PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 4 :		(11) International Publication Number:
G06F 11/10, H03M 13/02	A1	WO 89/ 02123
		(43) International Publicati n Date: 9 March 1989 (09.03.89)

(21) International Application Number: PCT/US88/02898

(22) International Filing Date: 23 August 1988 (23.08.88)

(31) Priority Application Numbers: 088,378
136,206

(32) Priority Dates: 24 August 1987 (24.08.87)
21 December 1987 (21.12.87)

(33) Priority Country: US

(71) Applicant: DIGITAL EQUIPMENT CORPORATION [US/US]; 146 Main Street, Maynard, MA 01754 (US).

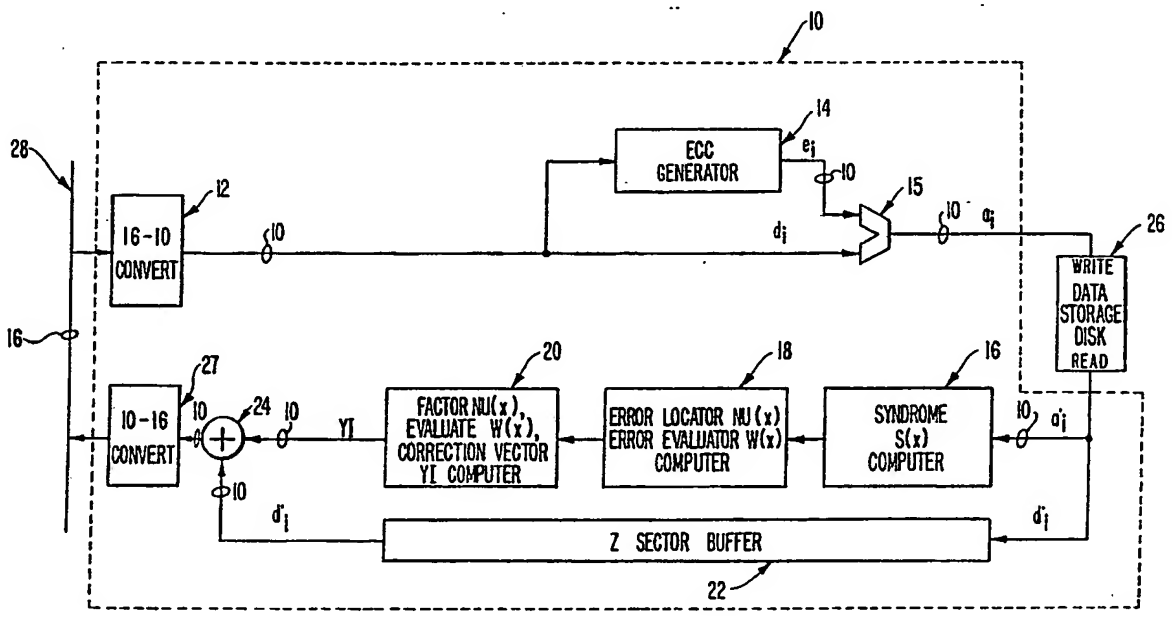
(72) Inventors: RIGGLE, Michael, C. ; 12995 Morris Trail, Colorado Springs, CO 80908 (US). WENG, Lih-Jyh ; 6 Bicentennial Drive, Lexington, MA 02173 (US). HUI, Pak, Ning ; East View Apt 97, 63 Frank Street, Worchester, MA 01604 (US).

(74) Agents: PATTERSON, Herbert, W.; Finnegan, Henderson, Farabow, Garrett & Dunner, 1775 K Street, N.W., Washington, DC 20006 (US) et al.

(81) Designated States: AT (European patent), BE (European patent), CH (European patent), DE (European patent), FR (European patent), GB (European patent), IT (European patent), JP, LU (European patent), NL (European patent), SE (European patent).

Published
With international search report.
Before the expiration of the time limit for amending the claims and to be republished in the event of the receipt of amendments.

(54) Title: HIGH BANDWIDTH REED-SOLOMON ENCODING, DECODING AND ERROR CORRECTING CIRCUIT



(57) Abstract

A VLSI circuit (10) that implements a symmetric Reed-Solomon error correcting code for correcting up to 32 symbol errors per code word in real time. Extensive use of data pipelining (16), (18), (20) and novel solutions to the Berlekamp-Massey algorithm without supplementary software or heavily iterative hardware calculations provide a high bandwidth, low latency implementation of the Reed-Solomon code with low data overhead and low hardware cost.

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AT	Austria	FR	France	ML	Mali
AU	Australia	GA	Gabon	MR	Mauritania
BB	Barbados	GB	United Kingdom	MW	Malawi
BE	Belgium	HU	Hungary	NL	Netherlands
BG	Bulgaria	IT	Italy	NO	Norway
BJ	Benin	JP	Japan	RO	Romania
BR	Brazil	KP	Democratic People's Republic of Korea	SD	Sudan
CF	Central African Republic			SE	Sweden
CG	Congo	KR	Republic of Korea	SN	Senegal
CH	Switzerland	LI	Liechtenstein	SU	Soviet Union
CM	Cameroon	LK	Sri Lanka	TD	Chad
DE	Germany, Federal Republic of	LU	Luxembourg	TG	Togo
DK	Denmark	MC	Monaco	US	United States of America
FI	Finland	MG	Madagascar		

HIGH BANDWIDTH REED-SOLOMON ENCODING, DECODING AND ERROR CORRECTING CIRCUIT

BACKGROUND OF THE INVENTION

I. FIELD OF THE INVENTION

This invention relates to using VLSI architecture for high bandwidth, low latency execution of the Reed-Solomon encode, decode, and error correction functions, capable of correcting a large number of symbol errors in binary digital data.

II. DESCRIPTION OF THE RELATED ART

Electronic communications and data processing systems transmit, receive and store electromagnetic signals representative of binary "zeros" and "ones." During data transfer along data channels, or during the process of storing data on and retrieving data from various magnetic, optical or other storage media, erroneous data values may occur. These errors may be the result of electromagnetic noise on the data channel or defects in the storage media. Isolated single bit random errors and contiguous multibit "bursts" of errors must be recognized and corrected using a system that is highly reliable, that is, one that does not produce many errors in the corrected data. At a minimum, the erroneous data must be recognized and flagged as erroneous rather than being acted upon.

Unlike data communications systems that have the luxury of retransmitting data until a error free transmission occurs, erroneous data stored on magnetic, optical or other storage media are permanently lost unless they can be corrected. By encoding data prior to storage or transmission, and decoding following reception or retrieval, errors may be detected and corrected prior to the data being released for subsequent use. Reed-Solomon codes are effective for the types of independent and bursty errors experienced on magnetic storage media. Information on encoding, decoding, and Reed-Solomon codes in particular may be found in "Error Correcting Codes" by Peterson and Weldon, The MIT Press, second edition (1972) and several other texts.

For binary data storage and retrieval systems, error correction begins prior to storing data on the applicable medium, here considered to be a magnetic disk. First the data are

encoded using an (N,K) Reed-Solomon code word of N m-bit symbols, consisting of K data symbols and N-K Error Correction Code (ECC) symbols. The ECC symbols are redundant and provide the information necessary to recognize erroneous data symbols and to reconstruct a maximum of T = (N-K)/2 data symbols.

A symbol is a series of m bits, for example, 6 or 10. The data stream of binary digits destined for storage on a disk sector usually consists of a series of 8, 16 or 32 bit words and must be coverted to a series of data symbols. The N symbols of the code word are representative of the coefficients of a polynomial in x of degree N-1 represented by a(x) where deg(a)=N-1. ECC symbols are generated such that each code word is evenly divisible by a generator polynomial, g(x), having N-K consecutive roots in the Galois Field, GF (2^m), and where each ECC symbol is an element α^i of GF(2^m). The generator polynomial, g(x) is chosen based on the extent and complexity of the error correction scheme, that is, how may symbols of detection and correction are desired. In an (N,K) code, deg(g)=N-K. If d(x) is a polynomial in x of degree K-1 with K data symbols d_i as coefficients, then the code word a(x) is:

$$a(x) = d(x) * x^{(N-K)} + e(x)$$

where

$$e(x) = \text{Remainder of } [d(x) * x^{(N-K)} / g(x)]$$

The coefficients of e(x) are the ECC symbols e_i. This process of dividing the data symbol polynomial d(x) by the generator polynomial g(x) ensures that each stored code word a(x) is evenly divisible by g(x). If a retrieved code word y(x) is error free, y(x) is equal to a(x) and is evenly divisible by g(x). If y(x) is not evenly divisible by g(x) then that code word is assumed to contain an error vector p(x) that must be located and corrected, where y(x) = a(x) + p(x).

The factors of an error locator polynomial, NU(x), an error evaluator polynomial, W(x), locate the erroneous data symbols within the code word and indicate the values which, when added modulo 2 to the received symbols, provide the correct symbols. The terms of NU(x) are calculated using the Berlekamp Massey algorithm from the error syndromes, Si, which are obtained by dividing y(x) by the factors of g(x). If a syndrome

is non-zero, then part of the code word is in error. The error located is found by computing the roots of the error located polynomial, NU(x) and then the correct data symbol may be determined by finding the value of W(x) using the syndromes and NU(x). Root finding in a digital environment typically uses Chien's search algorithm. A Reed-Solomon encoding-decoding system that uses the technique of a generator polynomial, syndrome computation, and evaluation of error locator and error evaluator polynomials is described in U.S. Patent No. 4,413,339 to Riggle et al.

Because of the many computations being performed in software or iteratively in hardware, and the need to calcuate the NU(x) and W(x) polynomials and all the roots of NU(x), and to evaluate W(x) prior to correcting erroneous data symbols, prior art decoders suffer from one or more of the drawbacks of high latency, low bandwidth and high cost, especially when many data errors are encountered. Soft error rates for disks are therefore limited to about 1 x 10-6 because of the low processing speed or cost of the decoders. Faster Reed-Solomon systems which are also inexpensive have the drawback of being limited to correcting a maximum of only about 4 symbols per sector, which limits the tolerable soft error rates for data storage devices. Cross-interleaved Reed-Solomon decoders have a moderate bandwidth and can correct large numbers of symbol errors, but they introduce excessive storage overhead for disk systems with short disk blocks, thereby reducing the net data density permitted on the disk drive. This defeats one of the primary purposes for implementing the ECC system, which is to allow high density data storage on disk drives along with the inherently high data error rates resulting from increased density data storage.

An object of this invention is to correct higher soft bit error rates on sectorized data storage devices while providing acceptably low hard bit error rates.

Another object of this invention is to provide a high bandwidth, low latency implementation of the Reed-Solomon error correction function where a large number of symbol errors are corrected with a minimum of data storage overhead within a standard disk block.

A further object of this invention is to implement pipelining and flow control processes for executing the Berlekamp-Massey, Weng and modified Chien algorithms, and to provide novel algorithms for computing the degree of the error location polynomial and its control over the Berlekamp-Massey algorithm, so that the Reed-Solomon error correction function can be implemented in hardware in real time.

Additional objects and advantages of the invention will be set forth in the description which follows, and in part will be obvious from the description, or may be learned by practice of the invention. The objects and advantages of the invention may be realized and obtained by means of the instrumentalities and combinations particularly pointed out in the appended claims.

Related Applications

U.S. Patent Application No. 050,725 titled "Real-Time BCH Error Correction Code Decoding Mechanism" by Lih-Jyh Weng filed on May 15, 1987, and assigned to the same assignee as the present invention, is hereby expressly incorporated by reference as though set out in full herein.

U.S. Patent Application No. 067,712 titled "Apparatus for Computing Multiplicative Inverses in Data Encoding and Decoding Devices" by Lih-Jyh Weng filed on June 26, 1987, and assigned to the same assignee as the present invention, is hereby expressly incorporated by reference as though set out in full herein.

SUMMARY OF THE INVENTION

To achieve the foregoing objects and in accordance with the purposes of the invention as embodied and broadly described herein, a circuit is provided for detecting and correcting errors in binary data symbols d_i that have passed through an error inducing medium, comprising encoder means for encoding the data symbols into a plurality of Reed-Solomon first code words a(x) and sending the first code words to the error inducing medium; buffer means for receiving from the error inducing medium a plurality of retrieved code words y(x) equal to the first code words in which errors have been induced and for holding each of the retrieved code words for no longer than

the time needed to compute error locations and values; decoder means for computing from the retrieved code words a plurality of error locations and correction vectors in real time as each symbol of the retrieved code word exists from the buffer means; and correcting means responsive to the error locations and correction vectors for correcting erroneous data symbols as they exit the buffer means.

The accompanying drawings, which are incorporated in and constitute a part of the specification, illustrate a preferred embodiment of the invention and, together with the general description given above and the detailed description of the preferred embodiment given below, serve to explain the principles of the invention.

BRIEF DESCRIPTION OF THE DRAWINGS

Fig. 1 is a schematic block diagram showing the overall structure of an integrated circuit which constitutes a preferred embodiment of the present invention;

Fig. 2 is an electrical schematic diagram showing the major steps performed by the ECC Generator of the circuit of Fig. 1;

Fig. 3 is a flow chart of the decoding process performed by the syndrome computer, the error locator NU(X) and error evaluator W(x) computer, and the factor NU(x), evaluate W(x), and correction vector YI computer circuits of Fig. 1;

Fig. 4 is an electrical schematic diagram of the Syndrome Compute circuit, part of the Snydrome Computer block 16 of the circuit of Fig. 1 and part of step 102 of the flowchart of Fig. 3;

Fig. 5 is an electrical schematic diagram of part of the Amend Syndrome Compute circuit, another part of the Syndrome Compute block 16 of the circuit of Fig. 1 and apart of step 102 of the flowchart of Fig. 3;

Fig. 6 is a flowchart of the computation of the error locator polynomial NU(x), step 104 of Fig. 3 and part of the Compute Error Locator NU(x), Error Evaluator W(x) circuit 18 of Fig. 1;

Fig. 7 is an electrical schematic diagram of the execution of Berlekamp's algorithm as shown in the flowchart of Fig. 6;

Fig. 8 is an electrical schematic diagram of the remaining part of the execution of Berlekamp's algorithm as shown in the flowchart of Fig. 6;

Fig. 9 is an electrical schematic diagram of a Galois Field combinational inverse calculator, which is circuit 364 of Fig. 7;

Fig. 10 is an electrical schematic diagram of the circuit that computes the error evaluator polynomial W(x), namely step 106 of Fig. 3;

Fig. 11 is an electrical schematic diagram of the circuit that computes the value of the error locator polynomial NU(x) in real time for each symbol of the code word, namely steps 108, 110, and 112 of Fig. 3;

Fig. 12 is an electrical schematic diagram of the circuit that computes the value of the error evaluator polynomial W(x) in real time for each symbol of the code word, namely step 114 of Fig. 3;

Fig. 13 is an electrical schematic diagram of the circuit that computes the error locations and correction vectors of code word symbols in real time, comprising the circuits of Figs. 11 and 12, and also including steps 116, 118, and 120 of Fig. 3.

Fig. 14 is an electrical schematic diagram of the Galois Field divide calculator, which is circuit 524 of Fig. 13;

Fig. 15 is a timing diagram showing the time allocations for calculations and data transfers in the circuit of the present invention;

Fig. 16 is an electrical schematic diagram of an .XOR. network that functions as a GF α^{-33} multiplier;

Fig. 17 is an electrical schematic diagram of an .XOR. network that functions as a GF α^{-32} multiplier.

Fig. 18 is an electrical schematic diagram of an .XOR. network known as a Shift-1 circuit that performs the function of a GF α^{1} multiplier;

Fig. 19 is an electrical schematic diagram of an .XOR. network known as a Shift-2 circuit that performs the function of a GF α^2 multiplier;

Fig. 20A is an electrical schematic diagram of a circuit to compute [SYMBOL A] x α^{-33} and [SYMBOL A] x α^{-32} using the α^{-33} multiplier of Fig. 16 and the α^1 multiplier of Fig. 18;

Fig. 20B is an electrical schematic diagram of a circuit to compute [SYMBOL B] X α^{30} AND [SYMBOL B] x α^{32} using an α^{30} multiplier and the Shift-2 circuit of Fig. 19;

Fig. 21 is an electrical schematic diagram of a preferred embodiment of the Syndrome Compute circuit of Fig. 4 using the Shift-1 circuit of Fig. 18 to eliminate approximately half of the multiplier circuitry;

Fig. 22A is a second preferred embodiment of the NU_{even} circuit Fig. 13 using the Shift-2 circuit of Fig. 19;

Fig. 22B is an electrical schematic diagram of a second preferred embodiment of the NU_{odd} circuit of Fig. 13 using the Shift-2 circuit of Fig. 19;

Fig. 23 is an electrical schematic diagram of a second preferred embodiment of the error evaluator polynomial W(x) computer of Fig. 12 using the Shift-1 circuit of Fig. 18;

DESCRIPTION OF THE PREFERRED EMBODIMENT

Reference will now be made in detail to the present preferred embodiment of the invention as illustrated in the accompanying drawings.

The present preferred embodiment of the Reed-Solomon error detecting and correcting circuit is shown in block diagram form in Figure 1 and is represented generally by the numeral 10. As embodied herein, circuit 10 accepts binary data from data bus 28 of a data processing system (not shown) and converts those data bits into K m-bit data symbols, d_i, where i = 1 to K. Data symbols d_i are simultaneously sent to an Error Correction Code (ECC) generator 14 and a data storage disk 26 through a multiplexer 15.

In accordance with the invention, there is provided encoder means for encoding the data symbols into a plurality of Reed-Solomon first code words a(x) and sending the first code words to the error inducing medium. As embodied herein, ECC

generator 14 generates N-K ECC symbols e_i, where i =0 to N-K-1, using a generator polynomial g(x) having roots α^i, $-34 < i < 34$, where α is a primitive element of the Galois Field $GF(2^m)$. N represents the length of the code word and is an integer $< 2^m$. The N-K ECC symbols e_i are appended by multiplexer 15 to the K data symbols d_i to create an N symbol Reed-Solomon (N,K) code word a(x), consisting of N symbols a_i, which is stored on data storage disk 26. A header field is also provided for each code word which provides separation between code words, and other storage functions, and is not described in detail. Storage device 26 is an example which contains an error inducing medium where errors can be created as data as being stored and retrieved.

Errors may occur to the symbols of code word a(x) when stored on and retrieved from storage device 26. When read from the disk, code word a(x) is designated y(x) because it may contain erroneous symbols. Where there are no errors present, y(x) equals a(x), or equivalently p(x) = 0.

In accordance with the invention, there is provided buffer means for receiving from the error inducing medium a plurality of retrieved code words y(x) equal to the first code words in which errors have been induced and for holding each of the retrieved code words long enough to compute error locations and values, preferably for no longer than the time needed to compute error locations and values. As embodied herein, data symbols read from data storage disk 26 are held in a data buffer 22. The invention further includes decoder means for computing from the retrieved code words a plurality of error locations and correction vectors in real time as each symbol of the retrieved code word exits from the buffer means. As herein embodied, the decoding means, including a Syndrome Computer 16, an Error Locator and an Error Evaluator Computer 18, and an Error Correction Vector Computer 20, calculates the error location and correction vector for data symbols being held in buffer 22.

As herein embodied, Syndrome Computer 16 begins calculating syndromes as the code word symbols are read for data storage device 26. Error locator and error evaluator computer 18 computes the error locator polynomial NU(x) by solving the

Berlekamp-Massey algorithm using syndromes computed in Syndrome Computer 16. The terms of error evaluator polynomial W(x) are also calculated in computer 18 using the values of NU(x) and syndromes from Syndrome Computer 16.

In accordance with the invention, there is provided correcting means responsive to the error locations and correction vectors for correcting erroneous data symbols as they exit the buffer means. As herein embodied, as each data symbol d'i is released from buffer 22, computer 20 calculates error locator polynomial NU(x) using a modified Chien search alogorithm, and evaluates error evaluator polynomial W(x) to compute a correction vector YI that is exclusive or'd by .XOR. circuit 24 with the data symbol being released from buffer 22. Data symbols of the decoded code word are converted into 16 bit data words by converter 27 and are returned to the data processing device 16 bit data bus 28.

The "pipe-lining" or high speed computation of error values and locations in real time as data are released from buffer 22 is a major factor in achieving the low latency, or low holdup time, which permits the circuit of the present invention to achieve a high bandwidth.

Using VLSI logic to execute a Reed-Solomon error correcting function, the circuit of the preferred embodiment achieves an acceptably low error rate for decoded data for sectorized data storage devices such as magnetic disks, magnetic tapes, and optical disks having high inherent error rates. The circuit provides a high bandwidth of approximately 70 megabits per second and a low latency of about 144 microseconds at peak bandwidth using a standard disk sector size of 480 symbols where up to 32 the bit symbols may need to be corrected, requiring an overhead of 670 density higher than certain earlier data storing systems.

Preferably, and as shown in Fig. 1, binary data bits destined for storage on magnetic disk 26 are received from a data processing system (not shown) via a 16 bit data bus 28. The binary data, which consist of both information words and Error Detecting Code (EDC) words are converted in converter 12 to a series of K symbols, where each symbol is m-bits long. In

the preferred embodiment, each symbol is 10 bits long, and K is selectable from 48 to 941 data symbols. The 10 bit symbols are assembled from data words and EDC words, each word being 16 bits long. The 16 bit words are converted to symbols until all data words and EDC words have been converted to ten bit data symbols, represented by d_i. Enough fill bits are added to provide complete symbols. Data symbols d_i are sent both to storage disk 26 and ECC generator 14.

As here embodied, the encoder means includes ECC generator 14. ECC symbols e_i which will be used to detect and correct errors in the data symbols d_i are computed by ECC generator 14 using a generator polynomial g(x). The error correction code chosen is a symmetric Reed-Solomon code over the Galois Field GF (2^{10}) generated by the primitive polynomial $x^{10} + x^3 + 1$. For the Reed-Solomon code of the preferred embodiment, the generator polynomial, g(x) is defined as:

Equation I

$$g(x) = \prod_{i=-33}^{+33} (x + \alpha^i),$$

where α^i are elements of $GF(2^{10})$. $GF(2^{10})$ contains 1023 elements, therefore a code word of N = 1023 symbols can be generated. Code words in the preferred embodiment are limited to N = 115 to 1008 symbols by the design of integrated circuit 10 of Fig. 1. The generator polynomial g(x) is chosen to generate 67 ECC symbols, therefore N-K is 67 which suffices to correct up to 32 symbols in error per sector or code word, provide a 100% probability of detecting all errors up to 35 errors per sector, and provide a strong likelihood of detecting all errors greater than 35 per sector. The polynomia g(x) is so defined that when expanded into its individual terms, it has symmetric coefficients so that the Galois Field multiplier circuits, defined mathematically below, can be reduced by a factor of 2, that is, each α^i term appears twice in g(x) as a coefficient of a different power of x. Equations 2 and 3 are expansions of Equation 1. Equation 3 shows the individual terms of g(x).

Equation 2

$$g(x) = (x + \alpha^{-32})\ (x + \alpha^{32})\ \ldots\ (x + \alpha^{33})$$

Equation 3

$$g(x) = x^{67} + \alpha^{35}x^{66} + \alpha^{146}x^{65} + \alpha^{572}x^{64} + \alpha^{47}x^{63} + \alpha^{505}x^{62} + \alpha^{950}x^{61} + \alpha^{820}x^{60} + \alpha^{812}x^{59} + \alpha^{177}x^{58} + \alpha^{259}x^{57} + \alpha^{530}x^{56} + \alpha^{444}x^{55} + \alpha^{974}x^{54} + \alpha^{349}x^{53} + \alpha^{255}x^{52} + \alpha^{261}x^{51} + \alpha^{898}x^{50} + \alpha^{1004}x^{49} + \alpha^{496}x^{48} + \alpha^{636}x^{47} + \alpha^{715}x^{46} + \alpha^{475}x^{45} + \alpha^{247}x^{44} + \alpha^{957}x^{43} + \alpha^{63}x^{42} + \alpha^{37}x^{41} + \alpha^{673}x^{40} + \alpha^{691}x^{39} + \alpha^{598}x^{38} + \alpha^{788}x^{37} + \alpha^{188}x^{36} + \alpha^{41}x^{35} + \alpha^{552}x^{34} + \alpha^{552}x^{33} + \alpha^{41}x^{32} + \alpha^{188}x^{31} + \alpha^{788}x^{30} + \alpha^{598}x^{29} + \alpha^{691}x^{28} + \alpha^{673}x^{27} + \alpha^{37}x^{26} + \alpha^{63}x^{25} + \alpha^{957}x^{24} + \alpha^{247}x^{23} + \alpha^{475}x^{22} + \alpha^{715}x^{21} + \alpha^{636}x^{20} + \alpha^{496}x^{19} + \alpha 1004x^{18} + \alpha^{898}x^{17} + \alpha^{261}x^{16} + \alpha^{255}x^{15} + \alpha^{349}x^{14} + \alpha^{974}x^{13} + \alpha^{444}x^{12} + \alpha^{530}x^{11} + \alpha^{259}x^{10} + \alpha^{177}x^{9} + \alpha^{812}x^{8} + \alpha^{820}x^{7} + \alpha^{950}x^{6} + \alpha^{505}x^{5} + \alpha^{47}x^{4} + \alpha^{572}x^{3} + \alpha^{146}x^{2} + \alpha^{35}x + 1$$

Fig. 2 shows the circuit used by ECC generator 14 to generate the 67 ECC symbols e_i by dividing a(x) by g(x). Mathematically, the code word a(x) is a polynomial with coefficients a_i and consists of the 67 ECC symbols e_i appended to the data symbols, d_i. Note that in the examples which follow, exactly 413 data symbols are chosen. Syndrome modifiers referenced later are based on this choice of 413 data symbols. If a(x) is considered to be a K-1 order polynomial in x with the symbols a_i as coefficients, then

Equation 4

$$a(x) * x^{N-K} = a_{N-1}x^{N-1} + a_{N-2}x^{N-2} \ldots + a_1x^{N-K+1} + a_0x^{N-K}$$

The coefficients of the first K terms are d_i, and the coefficients of the last N-K or 67 terms of the polynomial are zero and will be replaced by e_i. The ECC symbols are generated by dividing $d(x) * x^{N-K}$ by g(x), as shown by the circuit of Fig. 2, which amounts to dividing the incoming data stream d_i by g(x), as shown by the circuit of Fig. 2, which amounts to dividing the incoming data stream d_i by g(x). Circuit 14 includes 67 shift registers, labeled E(0), E(1) ... E(66), each of which is preferably a 10 bit parallel shift register, with the number in

parentheses being the power of x represented by that register. The data stream of 413 data symbols is read into ECC generator 14 as the data symbols are being sent to disk 26 for storage. The mode of generator 14 is then switched by blocking both the data gate 42 and the feedback gate 44 to transfer ECC symbols out of circuit 14 to be appended to the data symbols. Blocking the feedback gate also clears the ECC generator for the next sector's data. The ⊕ symbols, one of which is labeled 46 in Fig. 2, indicate a parallel .XOR. or modulo-2 add function. The x α^i symbols, one of which is labeled 48, indicate a Galois Field multiply by α^i, which is equivalent to raising the exponent of an input symbol by i, for example α^j x $\alpha^i = \alpha^{i+j}$, where i+j is computed modulo 1023. All GF multiply operations are modulo the primitive polynomial $x^{10} + x^3 + 1$.

Multiplication on the Galois Field is defined as follows. If p and q are any two 10 bit symbols where each bit is p0, p1, ... p9 and q0, q1, ... q9, then the GF multiply function is defined as:

Equation 5

$$p \times q = [(p9x^9 + p9x^8 \ldots p0x^0)(q9x^9 + p9x^8 \ldots q0^0)], \text{ modulo } x^{10} + x^3 + 1$$

$$= x^9[p9(q0 + q7) + p8(q1 + q8) + p7(q2 + q9) + p6q3 + p5q4 + p4q5 + p3q6 + p2q7 + p1q8 + p0q9] +$$

$$x^8[p9(q6 + q9) + p8(q0 + q7) + p7(q1 + q8) + p6(q2 + q9) + p5q3 + p4q4 + p3q5 + p2q6 + p1q7 + p0q8] +$$

$$x^7[\ldots$$

The coefficients of the remaining terms (x^7 to x^0) are the normal products of the polynomials, modulo $x^{10} + x^3 + 1$. Because the elements of p and q are binary digits, coefficients of the form pi(qj+qk) are equivalent to pi .AND. (q_j .XOR. q_k) and may be calculated in hardware in the normal manner. Any product is an .AND. and any addition in parentheses is a modulo-2 addition, of .XOR.

Because the Reed-Solomon code is cyclic, a 67 symbol "coset leader" is .XOR.'d with the 67 ECC symbols as they are shifted out of circuit 14. The coset leader prevents the decoding position of circuit 10 from misinterpreting and accepting a sector that has improperly been shifted. A shift of bits that have been .XOR.'d with the coset leader causes circuit 10 to interpret all data in that sector as containing >32 errors. During the data read operation, discussed hereinbelow, the 67 ECC symbols as read from the disk sector are again .XOR.'d with the same coset leader symbols, removing them from the ECC symbols and having no effect on further calculations unless the symbols were shifted, and then excessive errors will be generated in accordance with the normal functioning of the decoding circuit.

The coset leader is .XOR.'d to the ECC symbols in the normal manner. The 67 coset leader symbols of the present invention are selected from a code word of a Reed-Solomon code that contains the Reed-Solomon code of the present invention as a subset. As determined by experimentation, the code word from which the coset leader is selected must have no more than 67 non-zero symbols. There is no known method for selecting an optimum coset leader, but the coset leader of the preferred embodiment, shown in Table 1, will result in 33 or more errors if the symbols of the code word are shifted by up to 100 bits. Starting with the last symbol, the coset leader symbols in octal are:

TABLE 1

Symbol No.	Value	Symbol	Value	Symbol	Value
0	404	23	324	46	1125
1	1245	24	1722	47	1401
2	1026	25	1413	48	402
3	722	26	1002	49	1564
4	1347	27	777	50	1604
5	1147	28	1370	51	627
6	245	29	1526	52	776
7	1140	30	471	53	1275
8	457	31	1626	54	135
9	560	32	573	55	501
10	164	33	402	56	42
11	1407	34	1562	57	410
12	1647	35	743	58	1632
13	362	36	421	59	1313
14	326	37	1230	60	1223
15	401	38	55	61	531
16	567	39	1453	62	1745
17	314	40	233	63	1442
18	1532	41	257	64	777
19	1142	42	665	65	1632
20	707	43	1525	66	630
21	424	44	1415		
22	665	45	1301		

After a varying amount of data storage time determined by the disk user, code word y(x) is read from disk 26. Because of possible errors in code word symbols, a(x) is now represented by y(x). If there are no errors in y(x), then a(x) = y(x). The flow chart of Fig. 3 shows the sequence of events performed by the decoder portion of circuit 10 which includes syndrome generating means, step 102; the means for calculating error locator polynomial NU(x), step 104, and error evaluator polymonial W(x), step 106; and means for simultaneously calculating the value of the error locator polynomial NU(x), steps 108, 110, and 112, and the value of the error evaluator polynomial W(x), step 114, and means for correcting erroneous symbols in code word

y(x) to error free symbols a_i in real time as each root of NU(x) and value of W(x) is calculated, steps 116, 118, 120 and 122. On Fig. 3, N represents the code length in symbols and I represents the symbol number.

Syndrome computer 16 of Fig. 1 generates syndromes S_i, step 102 of Fig. 3, which are the coefficients of a polynomial S(x). Fig. 4 shows the circuit used to compute S(x) simultaneously as the symbols of code word y(x) are read from a sector of disk 26. The calculation being performed by the circuit of Fig. 4 is:

Equation 6

$$Sj = \sum_{i=0}^{N-1} Y_i \, x\alpha^{ij} \text{ for } -34 < j < 34$$

The calculation is performed by Galois Field multiplier means consisting of an array of 10 bit wide registers, labeled S_{33} to S_{-33}, GF multipliers, adders, and .AND. gates, each of which is an S computer. As in Fig. 2, x α^i symbols indicate a GF multiply operation, and 0 symbols indicate a modulo-2 addition, or .XOR. A mode switch occurs by deasserting feedback gate 150 after 479 symbols have been transferred into all 67 S computers, at which point all 67 .AND. gates, one of which is identified as 152, are blocked and the 480th symbol is simultaneously entered into the 67 S computers. Blocking feedback gate 150 clears all S computers in preparation for calculating syndromes for the next sector.

In a second preferred embodiment a substantial reduction in hardware was achieved by making the Galois Field multiplier means of the syndrome computer of Fig. 4 operate on a half symbol clock (70 ns.) even though it accepts input symbols at half that rate. This allows a hardware savings because each multiplier can be used twice in a full symbol period to generate two partial syndromes, therefore only 34 of the 67 alpha multipliers need be built. Each adjacent multiplier is related by a simple function. As an example, α^{-32} can be derived by taking α^{-33} and multiplying it by α^1 (-33 + 1 = -32). Multiplying by α^1 is equivalent to shifting once and XORing bit 9 with bit 2.

Given a 10 bit symbol 9 8 7 6 5 4 3 2 1 0, this symbol multiplied by α^1 would become 8 7 6 5 4 3 9.XOR.2 1 0 9. This is a much simpler and smaller function to build than an α^{-32} multiplier. All even numbered alpha multipliers in the syndrome computer are derived from the odd numbered alpha multipliers in this matter. This same hardware reduction technique is also used in the YI computer 20 of Fig. 1, and will be discussed in greater detail hereinbelow.

The 67 syndrome values S_i are next passed to the S'(K) registers of amend syndrome circuit 200 of Fig. 5. Except for syndrome value S_{33}, syndrome values are input through multiplexers, one of which is identified by the numeral 204. This releases the S calculator circuit of Fig. 4 to calculate syndromes for the next code word to be read. Circuit 200 is also part of syndrome generator 16 of Fig. 1. The amend syndrome function shortens the code symbols of $GF(2^{10})$ from a maximum of 1023 symbols to the 480 symbols used in the example and stores the amend syndromes for the shortened code in the SY registers, 210 of Fig. 5. Amend syndrome computation starts with a broadside load of the 67 syndromes, S(K), into S'(K) registers, labeled S'(-33) to S'(33). Simultaneously, register 202 is cleared and loaded with the value of α^{462}, which is equal to 0001010101. Multiplexers connected to S'(K) registers to the next higher S'(K-1) register. The first multiplication to take place in GF multiplier 206 is S(-33) x α^{462} which is then shifted into shift register SY(0). Multiplexer 208 isolates the α^{462} signal from registers 202 and aligns the GF x α^{544} multiplier so that when syndrome S(-32) is shifted into register S'(-33) it will be multiplied by the contents of register 202 which is the GF product of α^{462} and α^{544}. Each amend syndrome in register SY(K) is therefore the GF product of the syndrome value from the S'(K) register multiplied by $(\alpha^{462})(\alpha^{544})^{n-1}$. The x α^{544} function is the same GF multiplication defined by Equation 5, but may be implemented by the following .XOR. operations, where O_i are the output bits of a 10 bit symbol, and I_i are the input bits.

Table 2

$O_9 = I_0 \oplus I_1 \oplus I_2 \oplus I_3 \oplus I_5 \oplus I_6$

$O_8 = I_0 \oplus I_1 \oplus I_2 \oplus I_4 \oplus I_5$

$O_7 = I_0 \oplus I_1 \oplus I_3 \oplus I_4 \oplus I_9$

$O_6 = I_0 \oplus I_2 \oplus I_3 \oplus I_8$

$O_5 = I_1 \oplus I_2 \oplus I_7$

$O_4 = I_0 \oplus I_1 \oplus I_6$

$O_3 = I_0 \oplus I_5 \oplus I_9$

$O_2 = I_0 \oplus I_1 \oplus I_2 \oplus I_3 \oplus I_4 \oplus I_5 \oplus I_6 \oplus I_8 \oplus I_9$

$O_1 = I_0 \oplus I_1 \oplus I_2 \oplus I_3 \oplus I_4 \oplus I_5 \oplus I_7 \oplus I_8$

$O_0 = I_0 \oplus I_1 \oplus I_2 \oplus I_3 \oplus I_4 \oplus I_6 \oplus I_7$

The α^{462} and α^{544} multipliers of circuit 200 are programmable in the circuit of the preferred embodiment and must be changed if the code word length is altered from 480 symbols.

The means for generating the NU(x) error locator polynomial is shown in Fig. 1 as circuit 18. Amend syndromes, SY(K) are now used to compute NU(x) using the Berlekamp-Massey algorithm, as modified by Weng, and using a data flow technique as suggested by Riggle to increase processing speed, step 104 of Fig. 3, and shown more completely as the flow chart of Fig. 6.

The flow chart of Fig. 6 shows the individual steps performed during the calculation of NU(x). Fig. 6 is broken down into sections by clock cycles or states, described in more detail hereinbelow, to show operations performed simultaneously. For the sake of clarity, quantities calculated in one section of the chart are made available to other boxes in the chart, but connecting lines are not shown.

The initial conditions for Fig. 6 are shown in step 252. The quantity D in step 252 is the integer value of the code distance, N - K + 1 = 68. The quantity Y, calculated in step 260, is defined to be 0 when DU=0. A state machine (not shown) controls the process steps which are repeated 67 times. Referring also to Fig. 7, during state 1, multiplexers (not shown) connect 32 symbol wide GF multiplier 350 to 32 SY registers 210 (SY(1)-SY(32)) and 32 NU registers 352. The GF product is performed as defined by Equation 5 on the NU and SY registers. This completes step 254 of Fig. 6. The product, X(I) of Fig. 6, is shifted to not-shift registers (not shown), and is

exclusive OR'd in tree fashion with SY(0) by the ^{+}T function 354 of Fig. 7 until a single bit symbol, DU, appears to 10 bit bus 356. This completes step 256 of Fig. 6. The +T function is the concatenated .XOR. sum of all inputs. If there are n inputs I_j, then the output $O = I_0 \oplus I_1 \ldots \oplus I_n$. For maximum speed the .XOR.'s are treed as shown below.

Table 3

I_0 I_1 I_2 I_3 I_4 I_5 I_6 I_7 .

+ + + +

+ +

. . . . . .

During state 1, the single 10 bit symbol is clocked into DU registers 358 and is clocked to the output stage of registers 358 during state 2. Simultaneously, as shown in step 258 of Fig. 6, the counting register 360 of Fig. 7 containing counter variable LLMR is decremented by 1 and tested for a condition of being less than 0. If less than 0, gate 362 is enabled. As shown in step 260 of Fig. 6, during state 2 the value of DU in shift register 358 of Fig. 7, is shifted to the output and is used to (1) compute a new value of Y and an inversion process 364 of Fig. 7, described hereinbelow, (2) compute a new value of Q = DU ⊗ P, step 262 of Fig. 6 performed by GF multipler 366 of Fig. 7, and (3) test for DU = 0, logic test 368 of Fig. 7 and step 264 of Fig. 6. If DU = 0 and LLMR < 0 then EN is set as a flag via gate 362. Inversion operation 364 is performed according to a procedure developed by Lih-Jyh Weng, and is shown in Fig. 8.

The GF inverse circuit 364 of Fig. 7 is shown in more detail in Fig. 9. The circuit of Fig. 9 computes the inverse

α^{-n} of an input symbol α^n by raising the input symbol to 32 power and multiplying it by the input symbol lowered to the -33 power. The result of $\alpha^{32n} x\alpha^{-33n}$ is α^{-n}. To accomplish this a 10 bit symbol α^n is input to an α^{32} .XOR. network 400, with bitinputs I_1 and bit outputs O_i as shown below:

Table 4

$O_9 = I_2 \oplus I_3 \oplus I_4 \oplus I_9$

$O_8 = I_1 \oplus I_2 \oplus I_5 \oplus I_6 \oplus I_7 \oplus I_8 \oplus I_9$

$O_7 = I_7$

$O_6 = I_2 \oplus I_3 \oplus I_5$

$O_5 = I_1 \oplus I_3 \oplus I_4 \oplus I_6 \oplus I_7 \oplus I_9$

$O_4 = I_1 \oplus I_2 \oplus I_7 \oplus I_9$

$O_3 = I_2 \oplus I_6 \oplus I_7 \oplus I_8 \oplus I_9$

$O_2 = I_1 \oplus I_2 \oplus I_3 \oplus I_4 \oplus I_6 \oplus I_8$

$O_1 = I_1 \oplus I_4 \oplus I_6 \oplus I_7 \oplus I_8 \oplus I_9$

$O_0 = I_0 \oplus I_2 \oplus I_3 \oplus I_4 \oplus I_7$

Each $\oplus$ represents an .XOR. The value α^{32n} is multiplied by α^n to get α^{33n}. Because α^{33n} are elements of a subfield, $GF(2^5)$, of $GF(2^{10})$, there are only 31 possible non-zero values for any α^{33n}. Using a selected 5 of the 10 bits available (I_0, I_1, I_2, I_3, I_5), it is possible to construct a logic network as defined by Table 5 below which outputs the 31 inverses of α^{33n} (i.e., all α^{-33n}). O_I are the output bits of a α^{-33n} network that are combusted from the input bits I_i. The input bits are the 10 bits of the symbol α^{33n} and the output bits are the 10 bits of the symbol α^{33n} and the output bits are the 10 bits of the symbol α^{-33n}.

Table 5

$$O_9 = Z_1+Z_5+Z_6+Z_7+Z_9+Z_{10}+Z_{11}+Z_{12}+Z_{13}+Z_{16}+Z_{19}+Z_{20}+Z_{25}+Z_{27}+Z_{28}+Z_{30}$$

$$O_8 = Z_3+Z_5+Z_6+Z_8+Z_{10}+Z_{14}+Z_{15}+Z_{16}+Z_{18}+Z_{19}+Z_{20}+Z_{21}+Z_{22}+Z_{25}+Z_{28}+Z_{29}$$

$$O_7 = 0$$

$$O_6 = Z_4+Z_6+Z_7+Z_9+Z_{11}+Z_{15}+Z_{16}+Z_{17}+Z_{19}+Z_{20}+Z_{21}+Z_{22}+Z_{23}+Z_{26}+Z_{29}+Z_{30}$$

$$O_5 = Z_1+Z_3+Z_7+Z_8+Z_9+Z_{11}+Z_{12}+Z_{13}+Z_{14}+Z_{15}+Z_{18}+Z_{21}+Z_{22}+Z_{27}+Z_{29}+Z_{30}$$

$$O_4 = Z_1+Z_3+Z_4+Z_6+Z_8+Z_{12}+Z_{13}+Z_{14}+Z_{16}+Z_{17}+Z_{18}+Z_{19}+Z_{20}+Z_{23}+Z_{26}+Z_{27}$$

$$O_3 = Z_2+Z_5+Z_6+Z_{11}+Z_{13}+Z_{14}+Z_{16}+Z_{18}+Z_{22}+Z_{23}+Z_{24}+Z_{26}+Z_{27}+Z_{28}+Z_{29}+Z_{30}$$

$$O_2 = Z_1+Z_2+Z_3+Z_4+Z_5+Z_8+Z_{11}+Z_{12}+Z_{17}+Z_{19}+Z_{20}+Z_{22}+Z_{24}+Z_{28}+Z_{29}+Z_{30}$$

$$O_1 = Z_1+Z_2+Z_7+Z_9+Z_{10}+Z_{12}+Z_{14}+Z_{18}+Z_{19}+Z_{20}+Z_{22}+Z_{23}+Z_{24}+Z_{25}+Z_{26}+Z_{29}$$

$$O_0 = Z_0+Z_1+Z_3+Z_5+Z_9+Z_{10}+Z_{11}+Z_{13}+Z_{14}+Z_{25}+Z_{16}+Z_{17}+Z_{20}+Z_{23}+Z_{24}+Z_{29}$$

$Z_0=I_0\bar{I}_1\bar{I}_2\bar{I}_3\bar{I}_5$ $Z_1=\bar{I}_0\bar{I}_1I_2I_3I_5$ $Z_2=I_0I_1I_2I_3I_5$ $Z_3=\bar{I}_0\bar{I}_1I_2I_3\bar{I}_5$

$Z_4=\bar{I}_0\bar{I}_1\bar{I}_2I_3I_5$ $Z_5=\bar{I}_0\bar{I}_1\bar{I}_2I_3\bar{I}_5$ $Z_6=\bar{I}_0I_1\bar{I}_2\bar{I}_3\bar{I}_5$ $Z_7=I_0I_1I_2I_3\bar{I}_5$

$Z_8=I_0I_1\bar{I}_2I_3\bar{I}_5$ $Z_9=\bar{I}_0I_1I_2I_3I_5$ $Z_{10}=\bar{I}_0\bar{I}_1\bar{I}_2\bar{I}_3I_5$ $Z_{11}=I_0I_1I_2\bar{I}_3\bar{I}_5$

$Z_{12}=\bar{I}_0I_1I_2\bar{I}_3\bar{I}_5$ $Z_{13}=\bar{I}_0I_1\bar{I}_2I_3I_5$ $Z_{14}=I_0\bar{I}_1I_2\bar{I}_3\bar{I}_5$ $Z_{15}=I_0\bar{I}_1\bar{I}_2I_3\bar{I}_5$

$Z_{16}=I_0\bar{I}_1\bar{I}_2\bar{I}_3I_5$ $Z_{17}=I_0I_1\bar{I}_2I_3I_5$ $Z_{18}=I_0\bar{I}_1\bar{I}_2I_3I_5$ $Z_{19}=\bar{}^{TM}I_0I_1I_2\bar{I}_3I_5$

$Z_{20}=I_0\bar{I}_1I_2I_3I_5$ $Z_{21}=I_0I_1\bar{I}_2\bar{I}_3\bar{I}_5$ $Z_{22}=I_0I_1\bar{I}_2\bar{I}_3I_5$ $Z_{23}=\bar{}^{TM}I_0\bar{I}_1I_2\bar{I}_3I_5$

$Z_{24}=\bar{I}_0 I_1 \bar{I}_2 \bar{I}_3 I_5$ $Z_{25}=\bar{I}_0 \bar{I}_1 \bar{I}_2 I_3 \bar{I}_5$ $Z_{26}=I_0 \bar{I}_1 I_2 I_3 \bar{I}_5$ $Z_{27}=\bar{I}_0 \bar{I}_1 I_2 \bar{I}_3 \bar{I}_5$

$Z_{28}=\bar{I}_0 I_1 I_2 \bar{I}_3 I_5$ $Z_{29}=\bar{I}_0 I_1 I_2 I_3 \bar{I}_5$ $Z_{30}=I_0 I_1 I_2 \bar{I}_3 I_5$

The final operation to invert α^n is to multiply α^{32n} by α^{-33n} to yield α^{-n} according to equation 5, presented earlier.

If EN is set, during state 3 LLMR has its sign bit complemented, step 268 of Fig. 6, NR input contents are replaced by NU contents, step 270, and the value of P is replaced by the value of Y. If EN is not set, then NR is shifted upwards (towards higher order terms). Also during state 3, NU input contents are replaced with the NU outputs .XOR.'d with NR @ 0Q, step 274 of Fig. 6, and SY registers are shifted upwards, step 266. When this has been completed, one of 67 loops has been completed as counted by R in step 276 of Fig. 6. During the 67th loop, the SY register shift is omitted to prepare for the W(x) computation. A test of LLMR is made during each loop to assure that LLMR is < 0. If it is not, then there are more than 32 = T errors in the code word and the sector is not correctable. After all 67 steps have been performed, the final NU(I) are coefficients of the error locator polynomial NU(x) for $0 < I < (T+1)$, and NU(0)=1.

Circuit 448 of Fig. 10 includes means for calculating error evaluator polynomial W(x). As embodied herein, the means for computing the values of error evaluator polynomial W(x) comprises w registers 450, amend syndrome 54 registers 210, error located polynomial NV registers 352, multiplier 350, and treed adder 354 which executes the procedure shown in Fig. 3, including, specifically step 106. For simplicity, the calculation is shown in step 106 in an inverse form. The coefficients of W'(x) are the coefficients of W(x) only in reverse order. To illustrate:

Equation 7

If

$$p(x) = p_o + p_1X + p_2X^2 + \ldots p_kX^k$$

then

$$p'(x) = p_k + p_{k-1}X + p_{k-2}X^2 + \ldots p_oX^k$$

The same is true for S' (x) and NU' (x). The mode of the NU(x) computer changes to a W(x) computer mode, shown in Fig. 10. Multiplexers (not shown) are set to connect NU registers 352 and

SY registers 210 to GF multiple 350 and tree'd .XOR. function 354 throughout computation of W(x). Ten bit bus 356 is connected to register W(31) of W(I) shift registers 450.

Starting with a new state 1, the products of SY and NU are multiplied by GF multiplier 350 and are tree .XOR.'d by device 354, the operation of which have both been described hereinabove. A single 10 bit symbol is input to register W(31) from bus 356. During state 2, the contents of all W(I) registers are input to all W(I-1) registers via 32 ten bit symbol busses 452 (downshifting the W values) and all SY(I) values are shifted to SY(I+1) (upshifting the SY(I) values). States 1 and 2 are repeated until 32 cycles have been completed. The contents of W(I) registers 450 contain the coefficients of error evaluator polynomial W(x) and the NU(I) registers 352 contain the coefficients of the error locator polynomial NU(x). NU(I) and W(I) values are then transferred to the next functions, steps 108, 110, 112, and 114 of Fig. 3. Registers 210, 352, and 450 are clearly in preparation for computing NU(x) and W(x) for the next code word.

The circuits of Fig. 13 provide the means for simultaneously solving the error locator polynomial NU(x), for obtaining the corresponding error value from NU(x) and W(x), and for correcting erroneous symbols in code word y(x) to error free symbols a_i as each symbol is shifted out of buffer 22 of Fig. 1, which includes the steps of (1) solving each term of NU(x), summing the odd and even power terms into $NU_{odd}(x)$ and $NU_{even}(x)$ respectively, step 108 of Fig. 3, (2) simultaneously evaluating W(x), and (3) correcting errors in y(x) as indicated by the error locations and error values derived from the calculated values of $NU_{odd}(x)$ and W(x). In order to generate a correction symbol, or correction vector YI, which is non-zero coefficient of p(x) for a given code word symbol, the error value computation circuit 20 of Fig. 1 evaluates the 32 symbol values of NU(x) plus the lowest order coefficient of NU(x) which is always 1, and the 32 symbol values of W(x) for each code word symbol within the sector in real time at the same time as the corresponding symbol is shifted out of buffer 22 of Fig. 1. The results of these evaluations are then manipulated to generate the actual correction polynomial p(x).

To determine error locations, the usual practice is to use a Chien search, where all possible legitimate values of x are tested to find the roots of NU(x). Based on those roots, the erroneous symbols are identified, W(x) evaluated, and y(x) later corrected. However, the standard technique requires more latency than the present invention. To provide the advantage of low latency, the Chien search is modified to permit simultaneous evaluation of W(x), data flow is pipelined, and erroneous symbols are identified and corrected in real time as they exit buffer 22 of Fig. 1 on their way to data processing bus 28. This requires a Chien search as modified by the present invention to compute the value of NU(x) and W(x) as each symbol exits buffer 22, whereupon a decision is made as to whether the symbol is incorrect. If incorrect, the non-zero coefficient in p(x), computed from NU(x) and W(x), amends the erroneous symbol in .XOR. circuit 24.

Circuit 480 of Fig. 11, consisting of a plurality of Galois Field multiplier means, each containing a 10 bit shift register and a Galois Field multiplier network, evaluates NU (α^i) as each code symbol is timed to exit buffer 22 of Fig. 1, where i is the number of the symbol exiting buffer 22 of Fig. 1. During each clock cycle, the contents of each NU register NU_1 to NU_{32} is multiplied by the GF multiplier connected to the register, thereby testing for a root α^i of NU for each clock cycle. The contents of each NU register of circuit 480 is multiplied by its corresponding GF multiplier once for each corresponding data symbol. Values of NU for the even powers of x and values of NU for the odd powers of x are summed separately and are referred to as NU_{even} and NU_{odd}. This completes steps 110 and 112 of Fig. 3. NU_{odd} is required for subsequent error value calculations. At the same time as the NU(x) calculations, W(x) is also being evaluated by circuit 500 of Fig. 12 once for each symbol exiting buffer 22. In particular, circuit 500 of Fig. 12 computes $W(\alpha^i) \times \alpha^{34i}$ for each symbol, corresponding to step 116 of Fig. 3.

Looking to Fig. 13, circuit 480 is the same circuit as shown in Fig. 11, and circuit 500 is the same circuit as shown in Fig. 12. Summer 520 computes NU(x) as the sum of NU_{odd} and

NU_{even}. If NU(x) = 1 then the i^{th} symbol of the code word exiting buffer 22 is in error, and gate 522 enables one input of gates 528 and 530, corresponding to step 120 of Fig. 3. At the same time, and regardless of whether the i^{th} symbol is correct or not, divider circuit 524 computes the correct value, or correction vector of the i^{th} code word symbol. This is shown as YI in step 116 of Fig. 3 and will be discussed in more detail hereinbelow. Also at the same time, logic gate 526 checks for $NU_{odd} = 0$. If it is zero and $NU_{even} = 1$, then gate 528 sets a system flag indicating to data processing system 28 of Fig. 1 that the decoded output of circuit 10 is erroneous. The disposition of the erroneous data is let up to the data processing system. Also, if $NU_{odd} = 0$, correction vector YI will not be gated to .XOR. circuit 24 because gate 530 of Fig. 13 will be locked open. This corresponds to step 118 of Fig. 3. An error location found while $NU_{odd} = 0$ indicates a malfunction of the decoding process.

Divider circuit 524 of Fig. 13, shown in greater detail in Fig. 14, operates in a similar manner to the GF inverse circuit of Fig. 9 except that a second input and third multiplier have been added. Divider 524 calculates the correction vector YI, step 116 of Fig. 3, by dividing Z from step 114, which is represented by symbol α^m of Fig. 14, by NU_{odd}, which is represented by symbol α^n of Fig. 14. The resulting correction vector is therefore:

Equation 8

$$YI = W(\alpha i)(\alpha^{34i})$$

This completes step 116 of Fig. 3.

Circuit 24 of Fig. 1 amends erroneous code word symbols by .XOR.ing the data stream from buffer 22 with the correction vectors, step 122 of Fig. 3. When a symbol is not in error, gate 530 of Fig. 13 is open and the correction vector is zero, therefore .XOR. circuit 24 leaves the data symbol unchanged. The buffer data stream must be synchronized with the computation of error values so that the correction vectors match the correct data symbols.

Buffer circuit 22 of Fig. 1 is a large first-in-first-out storage unit for data read from disk data port 26 and provides data to amend circuit 24 approximately 2 sector times following data entry to the buffer. In the preferred embodiment the input symbols are latched into a shift register once for each symbol read time, generating a 30 bit word. Each group of 3 symbols shifted in is written to a holding register, then later transferred into the buffer where the group remains until the end of the two sector delay time, whereupon the group of 3 symbols is received by a read shift register and sent to the data amend function 24 of Fig. 1, one symbol at a time.

The final function performed by the decoding circuitry is conversion of the amended 10 bit symbols into 16 bit data words and EDC words by converter 27 of Fig. 1. Bits 0-9 of the first data word are bits 0-9 of the first symbol. Bits 10-15 of the second data word are bits 0-5 of the second symbol. Conversion continues for all data words so that bits 0-7 of symbol 413 are bits 8-15 of the last data word, and bits 8 and 9 of the 413th symbol are discarded.

In describing the operation of circuit 10 of Fig. 1, reference is made to Fig. 15 which shows the time allocations for major data shifts and computations performed by the encoding and decoding circuits. The data flow between sections of circuit 10 is unidirectional, that is, there is no handshaking. Each section is ready to accept new data when the previous section is ready to supply it. The chip manipulates data in a synchronous manner, having one single clock source. The cycle time of the clock in the preferred embodiment is nominally 70 ns which is half-symbol time. A full symbol time in the preferred embodiment is 140 ns.

Decoding commences with data symbols being read from the data storage device 26 into buffer 22 and syndrome computer 16 of Fig. 1, time slots 600 and 602 of Fig. 15, respectively. ECC symbols at time slot 604 are also read into syndrome computer 16. Buffer 22 holds data read during time slot 600 for approximately two sector times and does not release the data until time slot 606. As data in time slot 602 are being read into syndrome computer 16 of Fig. 1, the syndrome computer begins processing

data during time slot 608. Following syndrome and amend syndrome computation, error evaluator polynomial W and error locator polynomial NU are computed during time slots 610 and 612 and are used to calculate correction vectors YI at time slow 614 as data symbols are being released from buffer 22 in time slow 606.

After the syndrome computation during time slot 608 is complete, syndrome computer 16 is now ready to begin computations on the next code word at time slow 616, with syndromes for these data being computed during time slow 618, and error evaluator and error locator polynomials being computed during time slots 620 and 622 respectively. Correction vectors during time slot 624 are calculated for the data of time slot 616 as they are released from buffer 22 during time slot 626.

This timing diagram clearly shows the pipelining made possible by the improved algorithms and circuits of the present invention.

In a second preferred embodiment, a savings is made in hardware in the exclusive OR (.XOR.) networks used to implement the GF $x\alpha^i$ multiplications shown in Fig. 4 for the syndrome computer, Fig. 12 or Fig. 13 for the error evaluator polynomial W(x) computer, and Fig. 11 or Fig. 13 for the NU_{odd} and NU_{even} computers. To implement the $z\alpha^i$ multipliers of the present invention, it is necessary to build an .XOR. network, such as the α^{-33} multiplier shown in Fig. 16. The ⊕ symbols are .XOR. gates. The input is a 10 bit symbol [SYMBOL A] and the output is a 10 bit symbol [SYMBOL A] $x\alpha^{-33}$.

The size of this kind of .XOR. network is large, requiring 59 .XOR. gates for the α^{-33} multiplier of Fig. 16, and 60 .XOR. gates for the α^{-32} multiplier of Fig. 17. These large .XOR. networks are complex to implement at the VLSI level, therefore, it is difficult to build many of them in a small silicon integrated circuit. The second preferred embodiment of the present invention provides an improved Galois Field multiplier means which reduces the number and complexity of these GF multiplier circuits.

On the Galois Field, $\alpha^i \times \alpha^j = \alpha^{i+j}$. For example, α^{-32} may be generated by multiplying α^{-33} by α^1. A given α multiplier

may therefore be constructed from products of simpler multipliers. The circuit of Fig. 18 shows an α^1 multiplier circuit, or Shift-1 circuit, which uses one .XOR. gate to perform a modulo-2 addition (bit 9 .XOR. bit 2) to generate the output bit 3 and also uses some cross-connected input and output lines to transpose bits to provide the desired output. The circuit of Fig. 19 shows an α^2 multiplier circuit, or Shift-2 circuit, consisting of two .XOR. gates and cross-connected inputs and outputs to produce the desired output. The circuits of Fig. 20a and 20b show how the product of a complex α multiplier, α^{-33} or α^{30}, and a simple multiplier, α^1 or α^2, can produce an output that ordinarily would require two complex multipliers, thereby considerably reducing circuit complexity. In Fig. 20a, an α^{-33} and an α^1 multiplier produce an output consisting of the input symbol [SYMBOL A] times α^{-33} and also times α^{-32} for a net savings of 59 .XOR. gates, or one α^{-32} multiplier network. Similarly, as shown in Fig. 20b, the values of [SYMBOL B]Xα^{30} and [SYMBOL B] xα^{32} are generated using a α^{30} multiplier and a Shift-2 circuit, thereby eliminating an α^{32} multiplier.

The application of this technique to the GF multiplier means for the present invention eliminates almost half of the complex .XOR. networks required by the syndrome computer and error location and error evaluator computers. Fig. 21 shows the syndrome computer of Fig. 4 configured to according to the second preferred embodiment to exploit the technique of reducing the complexity of the GF multipliers. To simplify the drawings, only the S(-30) to S(-33) registers and related GF multiplier and multipliers (MUXes) are shown and described; the rest being repetitive. The function of the syndrome computers has already been described hereinabove. Only the application of the GF multiplier means hardware reduction techniques to the syndrome computer will now be discussed.

Syndrome computer 16 of Fig. 1 processes one symbol through the syndrome compute circuit of fig. 4 or Fig. 21 approximately every 140 ns (a full symbol clock) in both preferred embodiments. In the second preferred embodiment of the syndrome computer circuit as shown in Fig. 21, the syndrome compute circuit consists of a plurality of Galois Field multiplier means each

comprising two 10-bit registers coupled to modulo-2 adder by a 2-to-1 multiplexer, a hold register, a complex GF multiplier .XOR. network, and a simple Shift-1 multiplexer.

Before syndrome computer 16 begins doing calculations, all S registers, 702, 704, 706, and 708 of Fig. 21, are cleared by a signal on reset line 738. During the first half symbol clock or approximately 70 ns, the 2-to-1 multiplexers (2-to-1 MUXes) 710 and 712 are enabled to allow the contents in the odd numbered S registers, S(-33), S(-31), etc. to modulo-2 add their contents by GF addition in adders 714 and 716 to the 10 bit data symbols from disk 26 of Fig. 1. The resulting sums are stored in hold registers 718 and 720. The contents of the hold registers are next multiplied by the GF multipliers, α^{-31} and α^{-33}, 722 and 724, and the resulting product is loaded into S(-31) and S(-33) registers 704 and 708. The contents of S(-31) register 704 are [(previous S(-31) register contents) + (data symbol from disk 26)] x α^{-31}, and the contents of S(-33) register 708 are: [(previous S(-33) register contents) + (data symbol from disk 26)] xα^{-33}.

In the next half symbol clock of approximately 70 ns, the 2-to-1 MUXes 710 and 714 are enabled to select the contents in the even numbered S registers 702 and 706, only two of which are shown in Fig. 21, and add the contents of the even numbered S registers to the same ten-bit symbol present on the data-in bus from disk 26 in adders 714 and 716. The result is stored in hold registers 718 and 720. The contents in the hold registers 718 and 720 are again multiplied by the GF multipliers, α^{-31} and α^{-33}, 722 and 724 respectively of Fig. 21, the output values are next multiplied by the Shift-1 circuits 726 and 728, and the results are stored in registers 702 and 706. The contents of S(-30) register 702 are [(previous S(-30) register contents + (data symbol from disk 26)] xα^{-30} and the contents of S(-32) register 706 are [(previous S(-32) register contents) + (data symbol from disk)] xα^{-32}. This add and multiply calculation is repeated for the next ten-bit data symbol retrieved from the disk.

In the first preferred embodiment of the syndrome computer, an output value of S is provided from the output of each adder

associated with each S register as shown in Fig. 4. In the second preferred embodiment, one output line, for example 730 of Fig. 21, provides the output for two S registers, 706, 708. One value of S is output every half symbol clock to multiplexer 732 which directs the symbol to its respective S' register of Fig. 5 via lines 734 and 736.

Figs. 22a and 22b show the error locator polynomial NU(x) computer circuit 480 of Fig. 11 or Fig. 13 for computing NU_{odd} and NU_{even} configured to according to the second preferred embodiment to exploit the technique of reducing the complexity of the GF multipliers. Figs. 22a and 22b show only small portions of the computers to illustrate the implementation of the technique. The circuits of Figs. 22a and 22b perform the same functions as the equivalent circuits 480 of Fig. 11 or Fig. 13 in the first preferred embodiment, therefore the function of these circuits will not be discussed further. Only the application of the GF multiplier hardware reduction techniques to the error locator computer will not be discussed.

In the second preferred embodiment as shown in Figs. 22a and 22b, the NU_{even} and NU_{odd} computer consists of a plurality of Galois Field multiplier means, each comprising two 10-bit shift registers coupled to a hold register by a 2-to-1 multiplexer, a complex GF multiplier .XOR. network, and a simple Shift-2 multiplier for repetitively multiplying the contents of the shift registers by a value of α^i. Before the NU calculators start doing calculations, a set of ten-bit symbol values are loaded into all NU even numbered registers 750, 752 (only a representative two of which are labeled) and all NU odd numbered registers 771, 773 (only a representative two of which are labeled) from bus 456 connection now shown). During the first half symbol clock of approximately 70ns the 2-to-1 MUXes 754, 775 are enabled to copy the contents of the NU(30) and NU(29) registers, 750, 771 respectively, into the hold registers 756, 777. The contents of hold registers 756, 777 are then multiplied by α^{30} and α^{29} GF multipliers 758, 779. The outputs of GF multipliers 758, 779 are loaded back into the corresponding NU(30) and NU(29) registers, 750, 771. The contents of NU(30) register 750 is now equal to [previous NU(30) register

contents] α^{30} and the contents of NU(29) register 771 is [previous NU(29) register contents]Xα^{29}.

During the next half symbol clock the 2-to-1 MUXes 754, 775 are enabled to copy the contents in the NU(32) and NU(31) registers 752, 773 into their respective hold registers 756, 777. The contents of the hold registers are next multiplied by the same GF α^{30} and α^{29} multipliers, 758, 779. The output values of the GF multipliers 758, 779 are loaded back into the corresponding NU(32) and NU(31) registers 752, 773 after being multiplied by α^2 in the respective Shift-2 circuits 762, 783. Now the contents of NU(32) register 752 is equal to [previous NU(32) register contents] xα^{31}. This process is again repeated in the next symbol clock.

Fig. 23 shows the error evaluator polynomial W(x) computer circuit 500 of Fig. 12 or Fig. 13, configured to according to the second preferred embodiment to exploit the technique of reducing the complexity of the GF multipliers. Fig. 23 shows only a small portion of the computer to illustrate the implementation of the technique. The circuit of Fig. 23 performs the same functions as the equivalent circuit 500 of Fig. 12 or Fig. 13 in the first preferred embodiment, therefore, the function of this circuit will not be discussed further. Only the application of the GF multiplier hardware reduction techniques to the error evaluator computer will now be discussed.

Error evaluator circuit 800 consists of a plurality of Galois Field multiplier means, of the same form as the NU_{odd} and NU_{even} circuits of Figs. 22a and 22b, for repetitively multiplying the contents of the NU registers by α^i, and functions in a similar manner. Even values of W() are calculated and stored in the even W registers during one half symbol clock and odd values are calculated and stored in the odd W registers during the next half symbol clock. Only even numbered α multipliers are provided; the odd α multipliers are provided by the Shift-1 circuit.

Because only half of the W values are calculated and summed in treed adder 802 during each half symbol clock, accumulator 804 stores and adds the even and odd outputs of adder 802 to provide the full value of W() for computation of the correction vector as shown on Fig. 13.

By applying the technique of the second preferred embodiment to the syndrome computer, only 34 complex GF multiplier .XOR. networks are required rather than the 67 required to implement the algorithm in hardware in the first preferred embodiment. Likewise, this technique reduces the need for .XOR networks in the NU_{even}, NU_{odd}, and W computers from 64 to 32.

Additional advantages and modifications will readily occur to those skilled in the art. The invention it its broader aspects is, therefore, not limited to the specific details, representative apparatus and illustrative examples shown and described. Accordingly, departures may be made from such details without departing from the spirit or scope of applicants' general inventive concept.

WHAT IS CLAIMED IS:

1. A circuit for detecting and correcting errors in binary data symbols d_1 that have passed through an error inducing medium, comprising:

a. encoder means for encoding said data symbols into a plurality of Reed-Solomon first code words a(x) and sending said first code words to said error inducing medium;

b. buffer means for receiving from said error inducing medium a plurality of retrieved code words y(x) equal to said first code words in which errors have been induced and for holding each of said retrieved code words for no longer than the time needed to compute error locations and values;

c. decoder means for computing from said retrieved code words a plurality of error locations and correction vectors in real time with each data symbol of said retrieved code words as each symbol exits from said buffer means; and

d. correcting means responsive to said error locations and correction vectors for correcting erroneous data symbols in said retrieved code words as they are released from said buffer means.

2. The circuit of claim 1 wherein said encoder means encodes K of said data symbols into a plurality of symmetrical Reed-Solomon first code words consisting of N symbols over the Galois Field $GF(2^m)$ generated by a primitive polynomial of x and said first code words are generated by dividing a polynomial in x with coefficients d_i by a generator polynomial g(x) characterized by:

$$g(x) = \prod_{i=-n}^{th} (x + \alpha^i)$$

to create N-K error Correction Code symbols e_i.

3. The circuit of claim 2 wherein said first code word a(x) is a polynomial in x with N coefficients a_i where the first a_i equal d_i and the last N-K a_i and K is less than N.

4. The circuit of claim 1 wherein said decoder means includes:

a. means for computing N-K syndromes S_i for each of said retrieved code words as said retrieved code words are received from said error inducing medium;

b. means for simultaneously computing the values of an error locator polynomial NU(x) and an error evaluator polynomial $W(x) * X^{34}$ in real time for each symbol of said retrieved code word as said retrieved code word is released from said buffer means where:

$$NU(x)S(x) = W(x) \text{ modulo } x^t$$

and S(x) is a polynomial in x with coefficients S_i and t is the error correcting capability of said Reed-Solomon Code;

c. means for calculating a correction vector YI for each symbol of said retrieved code word as said retrieved code word is released from said buffer means where:

$$Yi = \frac{W(\alpha^i) \times \alpha^{34i}}{(NUodd\ (\alpha^i)}.$$

and where α^i is an element of $GF(2^m)$ and NUodd (α^i) is the sum of the odd power terms of NU(x) calculated at α^i.

5. The circuit of claims 3 or 4 wherein m = 10, n = 33, said primitive polynomial is $x^{10} + x^3 + 1$, N is less than 2^m, and N-K is 67, thereby providing for the correction of 32 ten-bit symbols of y(x).

6. The circuit of claim 1 wherein the preselected number of said retrieved code words is two.

7. The circuit of claim 2 wherein said encoder means further includes means to exclusive "or" a coset leader to 67 symbols with said Error Correction Code symbols e_i, and said decoder means further includes means to exclusive "or" said coset leader with said retrieved code word.

8. A method for detecting said correcting errors in binary data symbols d_i that have passed through an error inducing medium, comprising:

a. encoding said data symbols into a plurality of Reed-Solomon first code words a(x) and sending said first code words to said error inducing medium;

b. receiving from said error inducing medium a plurality of retrieved code words y(x) equal to said first code words in which errors have been induced and holding each of said retrieved code words for no longer than the time needed to compute error locatins and values;.

c. computing from said retrieved code words a plurality of error locations and corrections vectors in real time with each data symbol of said retrieved code word as it is released from said step of holding; and

d. correcting said erroneous data symbols in said retrieved code word by said correction vectors at the computed locations.

9. The method of claim 8 wherein the step of encoding includes encoding K of said data symbols into a plurality of symmetrical Reed-Solomon first code words consisting of N symbols over the Galois Field GF (2^m) generated by a primitive polynomial of x and said first code words are generated by dividing a polynomial in x with coefficients of d_i by a generator polynomial g(x) characterized by:

$$g(x) = \prod_{i=-n}^{tn} (x + \alpha i)$$

to create N-K Error Correction Code symbols e_i.

10. The method of claim 9 wherein said first code word a(x) is a polynomial in x with N coefficients a_i where the first a_i equal d_i and the last N-K a_i equal e_i and K is less than N.

11. The method of claim 8 wherein said step of computing includes:

a. computing N-K syndromes S_i for each of said retrieved code words as said retrieved code words are retrieved from said error inducing medium;

b. simultaneously computing the values of an error locator polynomial NU(x) and an error evaluator polynomial $W(x) * X^{34}$ in real time for each data symbol of said retrieved code word as said code word is released from said step of holding where:

$$NU(x)\ S(x) = W(x) \text{ modulo } x^t$$

and S(x) is a polynomial in X with coefficients S_i and t is the error correcting capability of said Reed-Solomon Code; and

c. calculating a correction vector YI for each data symbol of said retrieved code word as said retrieved code word is released from said step of holding where:

$$YI = \frac{W(\alpha^i) \times \alpha^{34i}}{NUodd\ (\alpha^i)}$$

and where α^i is an element of $GF(2^m)$ and NUodd (α^i) is the sum of the odd power terms of NU(x) calculated at α^i.

12. The method of claims 10 and 11 wherein m = 10, n = 33, said primitive polynomial is $x^{10} + x^3 + 1$, N is less than 2^m and N-K is 67, thereby providing for correction of 32 ten-bit symbols of y(x).

13. The method of claim 8 wherein said preselected number of said retrieved code words is two.

14. The method of claim 9 wherein said step of encoding further includes the step of exclusive "or-ing" a coset leader of 67 symbols with said Error Correction Code symbols e_i, and said step of decoding further including the step of exclusive "or-ing" said coset leader with said retrieved code word.

15. A circuit for detecting and correcting errors in binary data comprising:

a. Reed-Solomon encoding means for encoding error free binary data to produce code words, each having K data symbols and N-K Error Correction Code (ECC) symbols, said ECC symbols being elements in the Galois Field GF (2^m) for any integer m greater than 0, said ECC symbols being generated by encoding said K data symbols using a predetermined generator polynomial g(x) having roots α^i where α is a primitive element in $GF(2^m)$;

b. means for transmitting each said code word to an error inducing medium;

c. means for receiving from said error inducing medium a retrieved code word equal to each said transmitted code word in which errors have been induced and for holding each of said retrieved code words for a preselected time period;

d. means for computing data error syndromes from said retrieved code words;

e. means for computing from said syndrome an error locator polynomial NU(x) of the form $NU(x) = NU_{odd}(x) + NU_{even}(x) + 1$.

f. means for computing from said syndromes and said error locator polynomial an error evaluator polynomial W(x);

g. means for evaluating NU(x) and $W(x)^{34}$ for a succession of elements α^i of $GF(2^m)$; and

h. means for correcting each data symbol of said retrieved code word using a correction vector YI calculated from said values of NU(x) and $W(x) * X^{34}$.

16. The circuit of claim 15 wherein m = 10, n < 1024, K = 67 and the number of erroneous data symbols that may be corrected in said retrieved code word is 32.

17. The circuit of claim 15 wherein said generator polynomial is

$$g(x) = \prod_{i=-33}^{+33} (x + \alpha^i)$$

18. The circuit of claim 15 wherein said preselected time period is the time required to retrieve one code word and calculate said error locator polynomial.

19. The circuit of claim 15 wherein said means for computing NU(x) uses the Berlekamp-Massey alogorithm.

20. The circuit of claim 15 wherein said means for evaluating NU(x) uses a Chien search.

21. The circuit of claim 15 wherein said error correction vector is

$$YI = \frac{W(\alpha^i \times \alpha^{34i}}{NUodd\ (\alpha^i)}$$

22. A method for detecting and correcting errors in binary data comprising:

a. encoding error free binary data to produce Reed-Solomon code word consisting of K data symbols and N-K Error Correcting Code (ECC) symbols, said ECC symbols being elements in the Galois Field $GF(2^m)$ for any integer m > 0, said ECC symbols generated by encoding said K data symbols by a predetermined generator polynomial g(x) having roots α^i where α is a primitive element in the $GF(2^m)$;

b. sending said code word to an error inducing medium;

c. receiving from said error inducing medium a retrieved code word equal to said code word in which errors have been induced and for holding each of said retrieved code words for a preselected time period;

d. computing data error syndromes from said retrieved code words;

e. computing from said syndrome an error locator polynomial NU(x) of the form $NU(x) = NU_{odd}(x) + NU_{even}(x) + 1$;

f. computing from said syndromes and said error locator polynomial and error evaluator polynomial W(x);

g. evaluating NU(x) and $W(x) * X^{34}$ for a succession of elements α_i of $GF(2^m)$; and

h. correcting each data symbol of said retrieved code word using a correction vector YI calculated from said values of NU(x) and $W(x)*X^{34}$.

23. The method of claim 22 wherein m = 10, N< 1024; K = 67 and the number of erroneous data symbols that may be corrected in said retrieved code word is 32 in said step of encoding error of free binary data.

24. The method of claim 22 wherein said generator polynomial is

$$g(x) = \prod_{i=-33}^{+33} (x + \alpha^i)$$

25. The method of claim 22 wherein said preselected time period is the time required to retrieve one code word and calculate said error locator polynomial.

26. The method of claim 22 wherein said step of computing NU(x) uses the Berlekamp-Massey alogorithm.

27. The method of claim 22 wherein said step of evaluating NU(x) uses a Chien search.

28. The method of claim 22 wherein said error correction vector YI equals:

$$YI = \frac{W(\alpha^i) \times \alpha^{34i}}{NUodd\ (\alpha^i)}$$

29. The circuit of claim 1 wherein said decoding means includes Galois Field (GF) multiplier means for repetitively GF multiplying symbols by α^i, said GF multiplier comprising:

a. 2-to-1 multiplexer for multiplexing a first input signal at a first predetermined time and a second input signal at a second predetermined time to obtain a first output signal;

b. a hold register for storing the first output signal;

c. means for coupling the first output signal to the hold register;

d. a first Galois Field (GF) multiplier for multiplying the stored first output signal to produce a second output signal;

e. a first register with an output coupled to the first input of the multiplexer for storing the second output signal at the first predetermined time;

f. a second GF multiplier for multiplying the second output signal to produce a third output signal; and

g. a second register with an output coupled to the second input of the multiplexer for storing the third output signal at a second predetermined time.

30. The circuit of claim 29 wherein the means for coupling the first output signal includes a modulo 2 adder.

31. The circuit of claim 4 wherein the means for computing N-K syndromes comprises:

a. a plurality of S computers for generating the coefficients S_i of a syndrome polynomial S(x); and

b. an amend syndrome calculator for shortening the code symbols of the Galois Field (2^m), from value of m=10 to a value of m less than 10.

32. The circuit of claim 31 wherein each S computer includes a Galois Field (GF) multiplier means for repetitively GF multiplying an input signal by α^i, said multiplier means comprising:

a. a logical .AND. gate responsive to a feedback signal and the input signal for providing a first output signal;

b. a GF multiplier responsive to the first output signal for GF multiplying the first output signal at a predetermined time to obtain a second output signal;

c. a storage register for storing the second output signal; and

d. a GF modulo-2 for adding the stored second output signal to a data signal and providing the sum as the input signal to the logical .AND. gate.

33. The circuit of claim 31 wherein each S computer includes a Galois Field (GF) multiplier means for repetitively multiplying a signal by α^i, said multiplier means comprising:

a. 2-to-1 multiplexer for multiplexing a first input signal at a first predetermined time and a second input signals at a second predetermined time to obtain a first output signal;

b. modulo-2 adder means for adding the first output signal to a data signal to obtain a second output signal;

c. a hold register for storing the second output signal;

d. a first Galois Field (GF) multiplier for multiplying the stored second output signal to produce a third output signal;

e. a first register with an output coupled to the first input of the multiplexer for storing the third output signal at the first predetermined time;

f. a second GF multiplier for multiplying the third output signal to produce a fourth output signal; and

g. a second register with an output coupled to the second input of the multiplexer for storing the fourth output signal at a second predetermined time.

34. The circuit of claim 4 wherein the means for computing the values of the error locator polynomial NU(x) includes Galois Field (GF) multiplier means for repetitively multiplying a signal by α^i, said multiplier means comprising:

a. storage means having an input port for receiving an input signal and an output port; and

b. a GF multiplier to recursively generate the input signal by GF multipplying the stored input signal from the output port.

35. The circuit of claim 4 wherein the means for computing the values of an error evaluator polymomial $W(x)* X^{34}$ includes Galois Field (GF) multiplier means for repetitively multiplying a signal by α^i, said multiplier means comprising:

a. storage means having an input port for receiving an input signal and an output port; and

b. a GF multiplier to recursively generate the input signal by GF multiplying the stored input signal from the output port at a predetermined time.

36. The circuit of claim 4 wherein the means for computing the values of an error locator polynomial NU(x) includes Galois Field (GF) multiplier means for repetitively multiplying a signal by α^i, said multiplier means comprising:

a. 2-to-1 multiplexer for multiplexing a first input signal at a first predetermined time and a second input signal at a second predetermined time to obtain a first output signal;

b. a hold register for storing the first output signal;

c. means for coupling the first output signal to the hold register;

d. a first Galois Field (GF) multiplier for multiplying the stored first output signal to produce a second signal;

e. a first register with an output coupled to the first input of the multiplexer for storing the second output signal at the first predetermined time;

f. a second GF multiplier for multiplying the second output signal to produce a third output signal; and

g. a second register with an output coupled to the second input of the multiplexer for storing the third output signal at the second predetermined time.

37. The circuit of claim 4 wherein the means for computing the values of an error evaluator polynomial $W(x)* X^{34}$ includes Galois Field (GF) multiplier means for repetitively multiplying a signal by α^i, said multiplier means comprising:

a. 2-to-1 multiplexer for multiplexing a first input signal at a first predetermined time and a second input signal at a second predetermined time to obtain a first output signal;

b. a hold register for storing the first output signal.

c. means for coupling the first output signal to the hold register;

d. a first Galois Field (GF) multiplier for multiplying the stored first output signal to produce a second output signal;

e. a first register with an output coupled to the first input of the multiplexer for storing the second output signal at the first predetermined time;

f. a second GF multipler for multiplying the second output signal to produce a third output signal; and

g. a second register with an output coupled to the second input of the multiplexer for storing the third output signal at the second predetermined time.

38. The method of claim 8 wherein the step of decoding includes Galois Field (GF) multiplying comprising:

a. multiplexing a first input signal at a first predetermined time and a second input signal at a second predetermined time to obtain a first output signal;

b. coupling the first output signal to a hold register;

c. storing the first output signal in the hold register;

d. first Galois Field (GF) multiplying of the stored first output signal to produce a second output signal;

e. storing the second output signal at the first predetermined time in a first register;

f. second GF multiplying the second output signal to produce a third output signal;

g. storing the third output signal at the second predetermined time in a second register;

h. coupling the stored second output signal to the first input of the multiplexer; and

i. coupling the stored third output signal to the second input of the multiplexer.

39. The method of claim 38 wherein the step of coupling includes the step of modulo-2 adding a data signal to the first stored output signal.

40. The method of claim 11 wherein the step of computing N-K syndromes comprises:

a. generating the coefficients of S_i of a syndrome polynomial S(x) in a plurality of S computers; and

b. shortening the code symbols of the Galois Field (2^m) in an amend syndrome calculator, from m = 10 to a value of m less than 10.

41. The method of claim 40 wherein the step of generating each coefficient of the syndrome polynomial S(x) comprises:

a. providing a first output signal from a logical AND gate responsive to a feedback signal and an input signal;

b. GF multiplying the first output signal at a predetermined time by a GF multiplier to obtain a second output signal;

c. storing the second output signal in a storage register; and

d. adding the stored second output signal to a data signal in a modulo-2 adder and providing the sum as the input signal to the logical .AND. gate.

42. The method of claim 40 wherein the step of generating each coefficient S of the syndrome polynomial S(x) comprises:

a. multiplexing a first input signal at a first predetermined time and a second input signal at a second predetermined time in a 2-to-1 multiplexer to obtain a first output signal;

b. adding the first output signal to a data signal in a modulo-2 adder to obtain a second output signal;

c. storing the second output signal in a hold register;

d. multiplying the stored second output signal in a GF multiplier to produce a third output signal;

e. storing the third output signal at the first predetermined time in a first register having an output coupled to the first input of the multiplexer;

f. multiplying the third output signal by a second GF multiplier to produce a fourth output signal; and

g. storing the fourth output signal at a second predetermined time in a second register having an output coupled to the second input of the multiplexer.

43. The method of claim 11 wherein the step of computing the value of the error locator polynomial NU(x) includes Galois Field (GF) multiplying comprising:

a. storing an input signal in a storage means having an input port and an output port; and

b. recursively GF multiplying the stored input signal from the output port by a GF multiplier to generate the input signal.

44. The method of claim 11 wherein the step of computing the values of an error evaluator $W(x)* X^{34}$ includes Galois Field (GF) multiplying comprising:

a. storing an input signal in a storage means having an input port and an output port; and

b. recursively GF multiplying the stored input signal from the output port by a GF multiplier to generate the input signal.

45. The method of claim 11 wherein the step of computing the values of an error locator polynomial NU(x) includes Galois Field (GF) multiplying comprising:

a. multiplexing in a 2-to-1 multiplexer a first input signal at a first predetermined time and a second input signal at a second predetermined time to obtain a first output signal;

b. coupling the first output signal to the hold register;

c. storing the first output signal in a hold register;

d. multiplying the stored first output signal by a first GF multiplexer to produce a second output signal;

e. storing the second output signal at the first predetermined time in a first register with an output coupled to the first input of the multiplexer;

f. multiplying the second output signal by a second GF multiplier to produce a third output signal; and

g. storing the third output signal at a second predetermined time in a second register coupled to the second input of the multiplexer.

46. The method of claim 11 wherein the step of computing the value of an error evaluator polynomial $W(x)*X^{34}$ includes Galois Field GF multiplying comprising:

a. multiplexing in a 2-to-1 multiplexer a first input signal at a first predetermined time and a second input signal at a second predetermined time to obtain a first output signal in a numeral 2-to-numeral 1 multiplexer;

b. coupling the first output signal to the hold register;

c. storing the first output signal in a hold register;

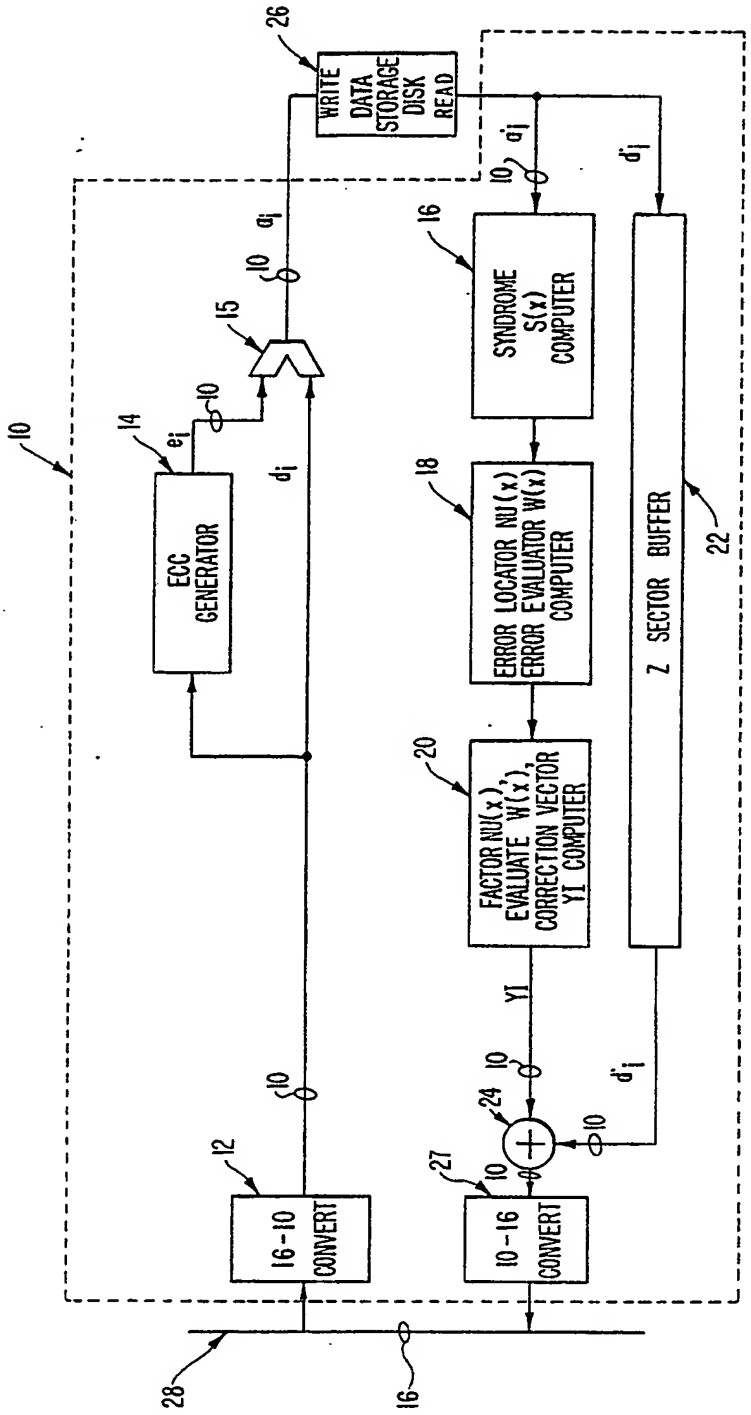
d. multiplying the stored first output signal by a first GF multiplier to produce a second output signal;

e. storing the second output signal at the first predetermined time in a first register with an output coupled to the first input of the multiplexer;

f. multiplying the second output signal by a second GF multiplier to produce a third output signal; and

g. storing the third output signal at a second predetermined time in a second register coupled to the second input of the multiplexer.

FIG. 1


26
WRITE DATA STORAGE DISK READ
10
16
SYNDROME S(x) COMPUTER
18
ERROR LOCATOR NU(x) ERROR EVALUATOR W(x) COMPUTER
20
FACTOR NU(x), EVALUATE W(x), CORRECTION VECTOR YI COMPUTER
22
2 SECTOR BUFFER
15
14
ECC GENERATOR
12
16-10 CONVERT
24
27
10-16 CONVERT
28
16

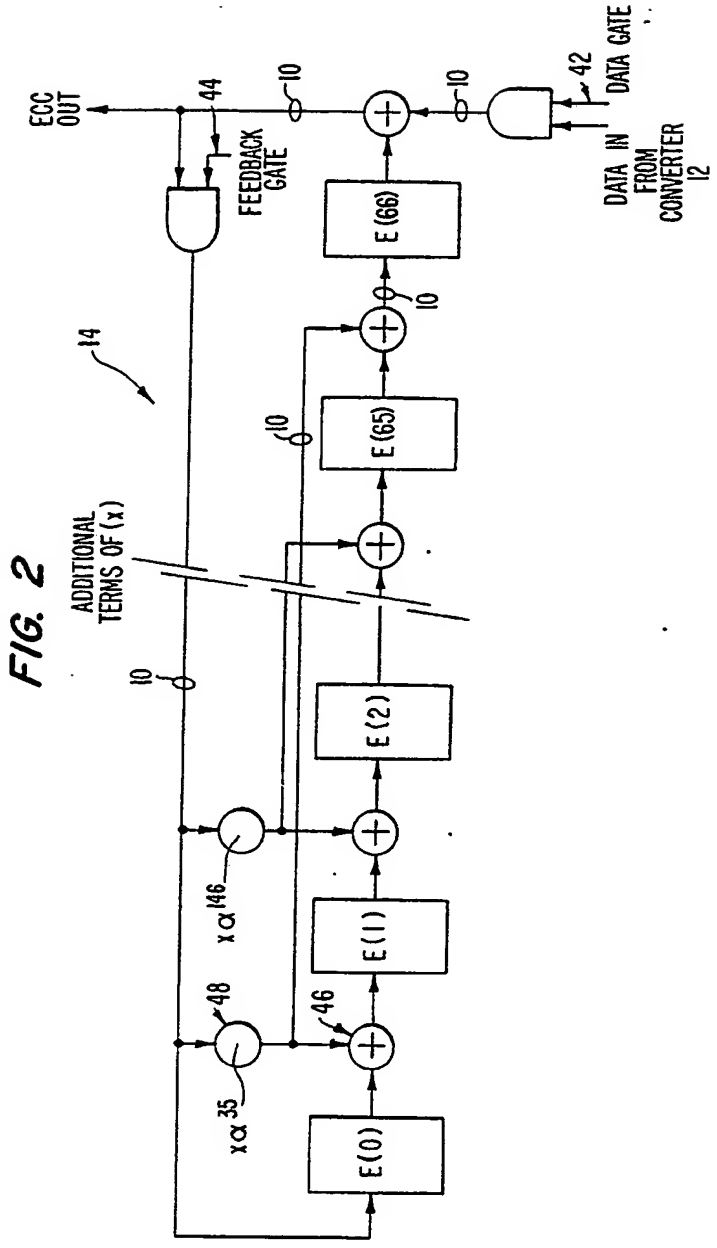

ECC OUT
44
10
10
42
DATA GATE
DATA IN FROM CONVERTER 12
FEEDBACK GATE
E (66)
10
14
E (65)
10
ADDITIONAL TERMS OF (x)
E (2)
10
$x\alpha^{146}$
E (1)
48
46
$x\alpha^{35}$
E (0)

FIG. 2

FIG. 3

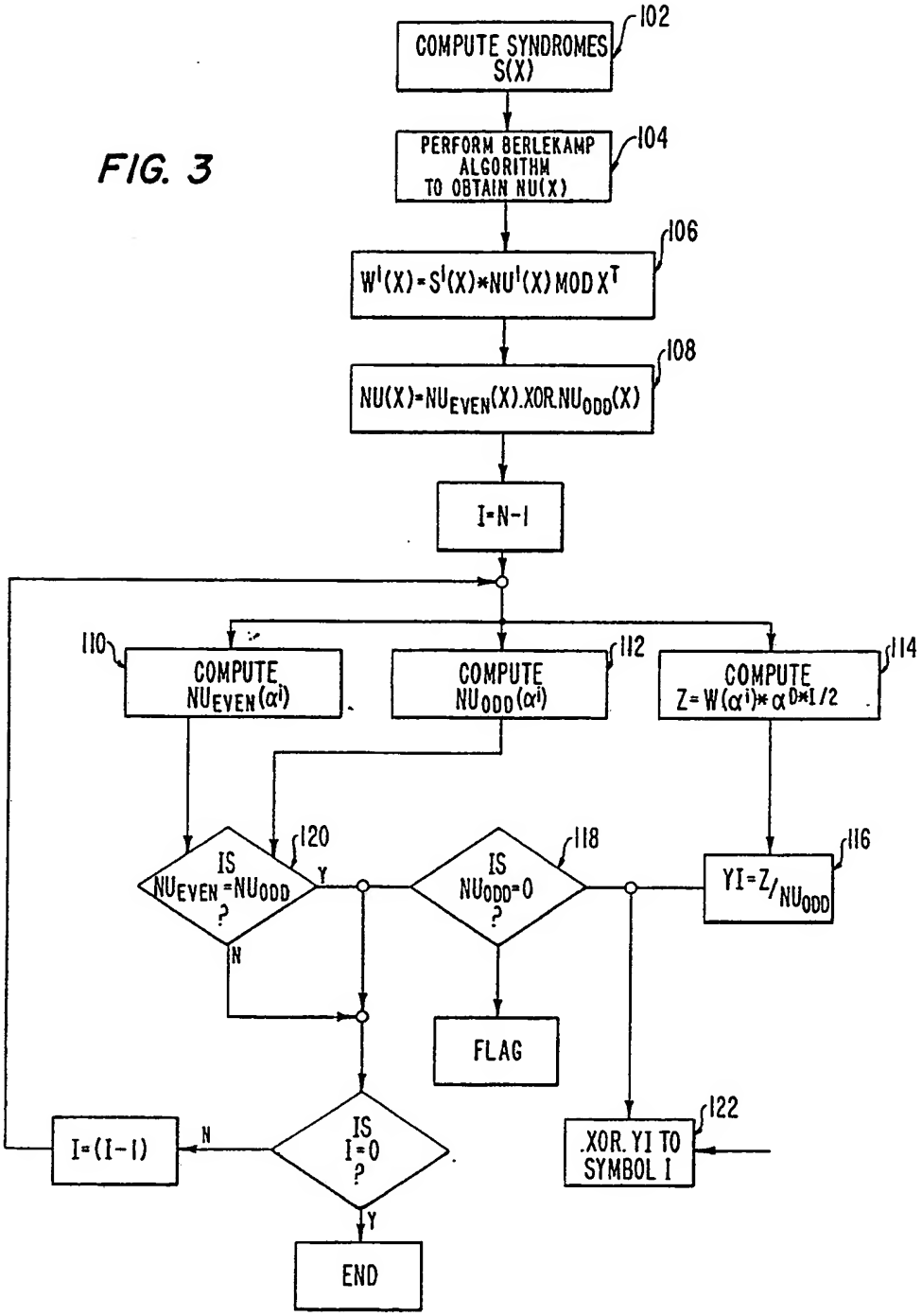

COMPUTE SYNDROMES S(X)
102
PERFORM BERLEKAMP ALGORITHM TO OBTAIN NU(X)
104
W^I(X)=S^I(X)*NU^I(X) MOD X^T
106
NU(X)=NU_EVEN(X).XOR.NU_ODD(X)
108
I=N-1
110
COMPUTE NU_EVEN(α^i)
112
COMPUTE NU_ODD(α^i)
114
COMPUTE Z=W(α^i)*α^(D*I/2)
120
IS NU_EVEN=NU_ODD ?
Y
N
118
IS NU_ODD=0 ?
116
YI=Z/NU_ODD
FLAG
IS I=0 ?
N
I=(I-1)
Y
122
.XOR. YI TO SYMBOL I
END

FIG. 4

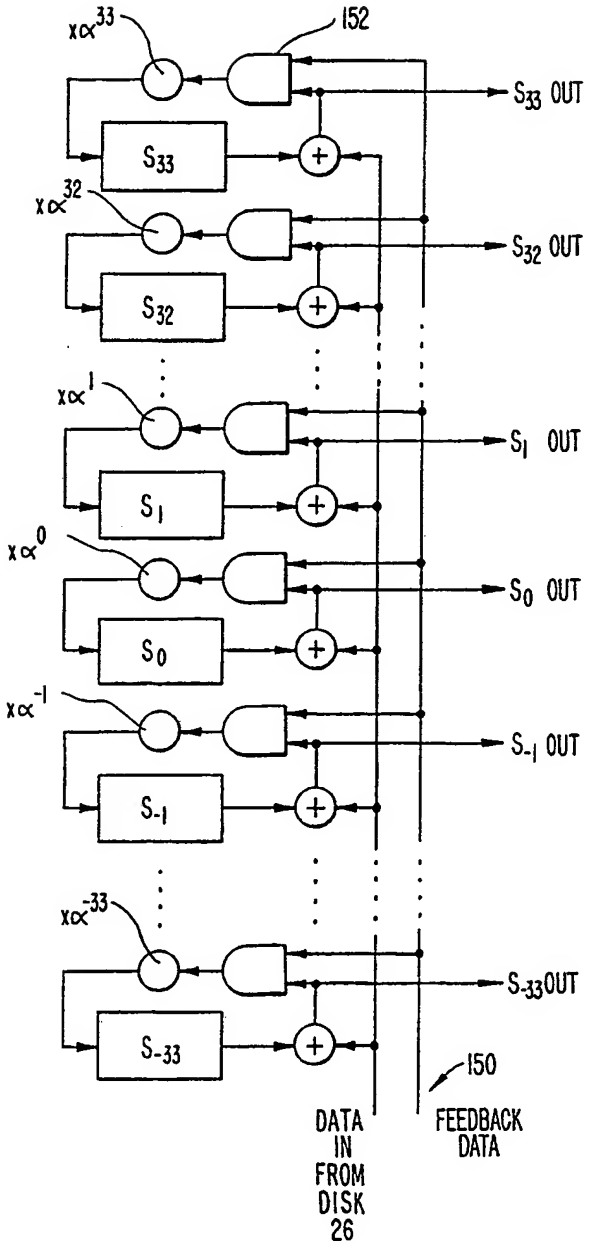

$x\alpha^{33}$
152
S_{33} OUT
S_{33}
$x\alpha^{32}$
S_{32} OUT
S_{32}
$x\alpha^{1}$
S_1 OUT
S_1
$x\alpha^{0}$
S_0 OUT
S_0
$x\alpha^{-1}$
S_{-1} OUT
S_{-1}
$x\alpha^{-33}$
S_{-33}OUT
S_{-33}
150
DATA IN FROM DISK 26
FEEDBACK DATA

FIG. 5

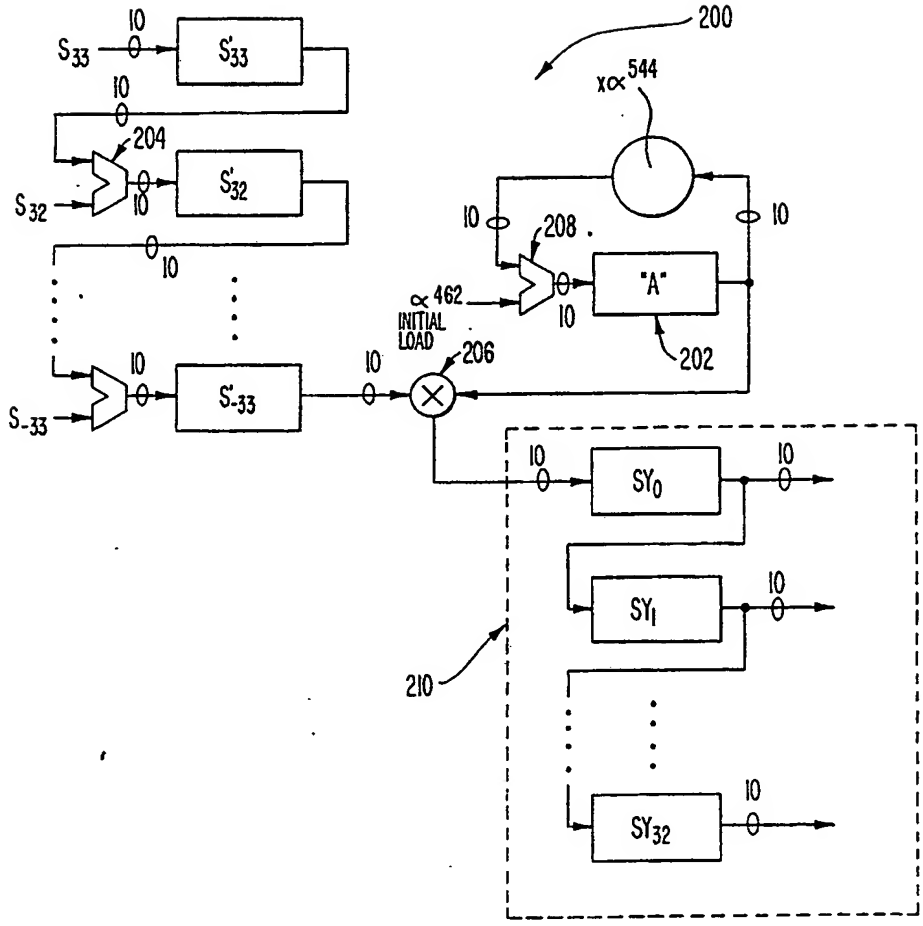

200
544
208
"A"
202
462
INITIAL LOAD
206
210
S_{33}
S'_{33}
S_{32}
S'_{32}
S_{-33}
S'_{-33}
SY_0
SY_1
SY_{32}
10

FIG. 6

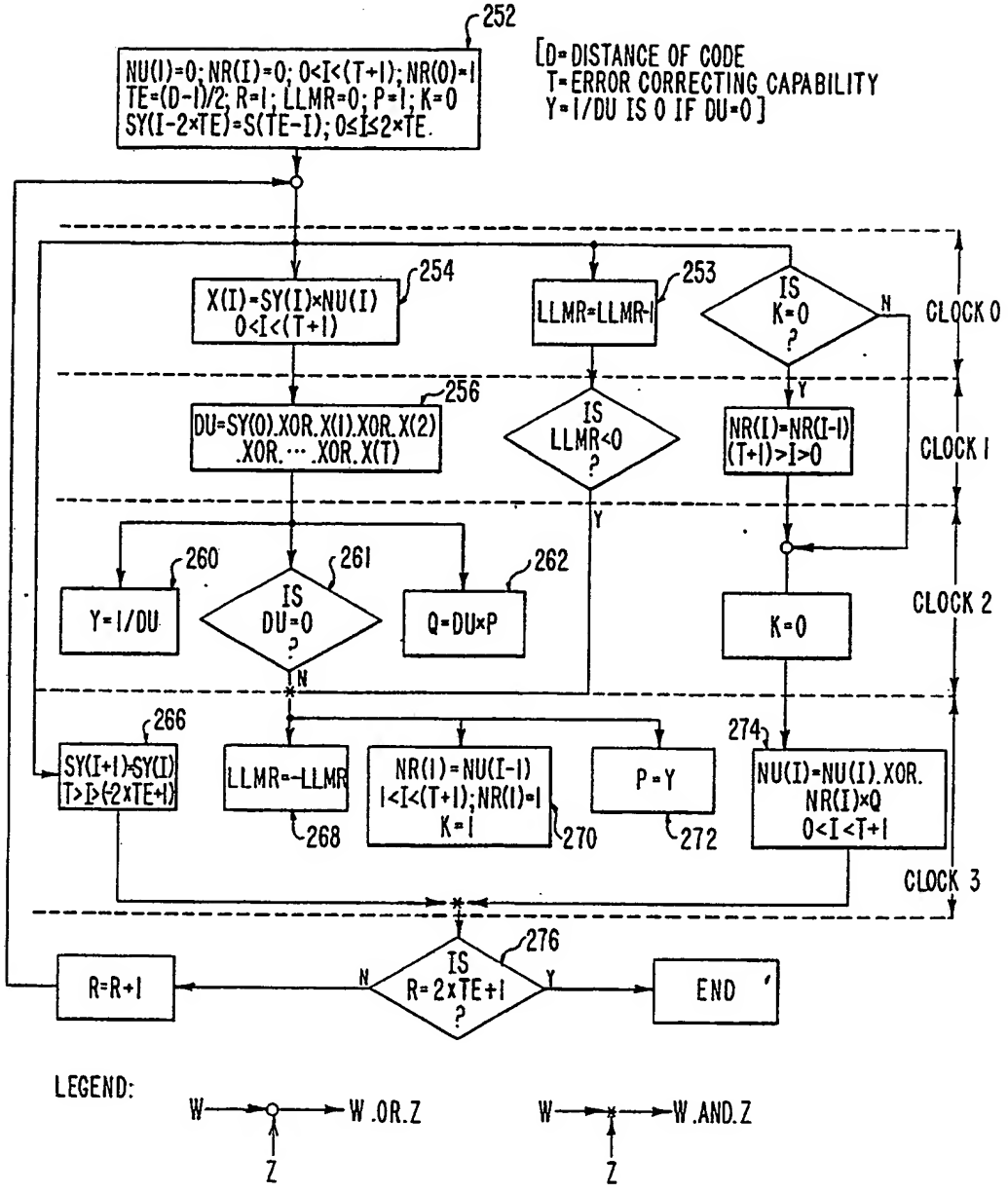
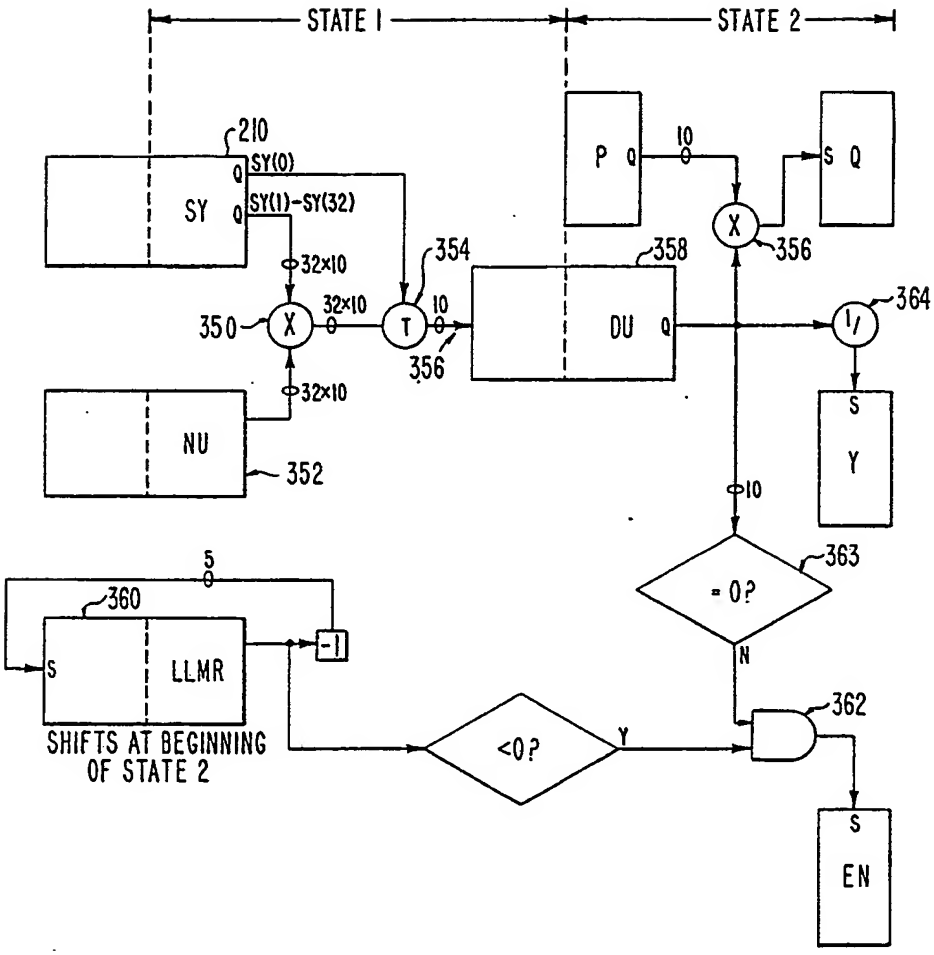

252
NU(I)=0; NR(I)=0; 0<I<(T+1); NR(0)=1
TE=(D-1)/2; R=1; LLMR=0; P=1; K=0
SY(I-2×TE)=S(TE-I); 0≤I≤2×TE.
[D=DISTANCE OF CODE
T=ERROR CORRECTING CAPABILITY
Y=1/DU IS 0 IF DU=0]
254
X(I)=SY(I)×NU(I)
0<I<(T+1)
253
LLMR=LLMR-1
IS K=0 ?
N
Y
CLOCK 0
256
DU=SY(0).XOR.X(1).XOR.X(2) .XOR.··· .XOR.X(T)
IS LLMR<0 ?
NR(I)=NR(I-1)
(T+1)>I>0
CLOCK 1
260
Y=1/DU
261
IS DU=0 ?
262
Q=DU×P
K=0
CLOCK 2
266
SY(I+1)=SY(I)
T>I>(-2×TE+1)
LLMR=-LLMR
268
NR(I)=NU(I-1)
1<I<(T+1); NR(I)=1
K=1
270
P=Y
272
274
NU(I)=NU(I).XOR.
NR(I)×Q
0<I<T+1
CLOCK 3
276
IS R=2×TE+1 ?
N
Y
R=R+1
END
LEGEND:
W
W .OR.Z
Z
W
W .AND.Z
Z

FIG. 7


STATE 1
STATE 2
210
SY(0)
SY(1)-SY(32)
SY
32×10
354
358
P
10
S Q
X
356
350
32×10
T
10
DU
364
356
NU
352
S
Y
10
363
= 0?
5
360
-1
S
LLMR
SHIFTS AT BEGINNING OF STATE 2
<0?
Y
N
362
S
EN

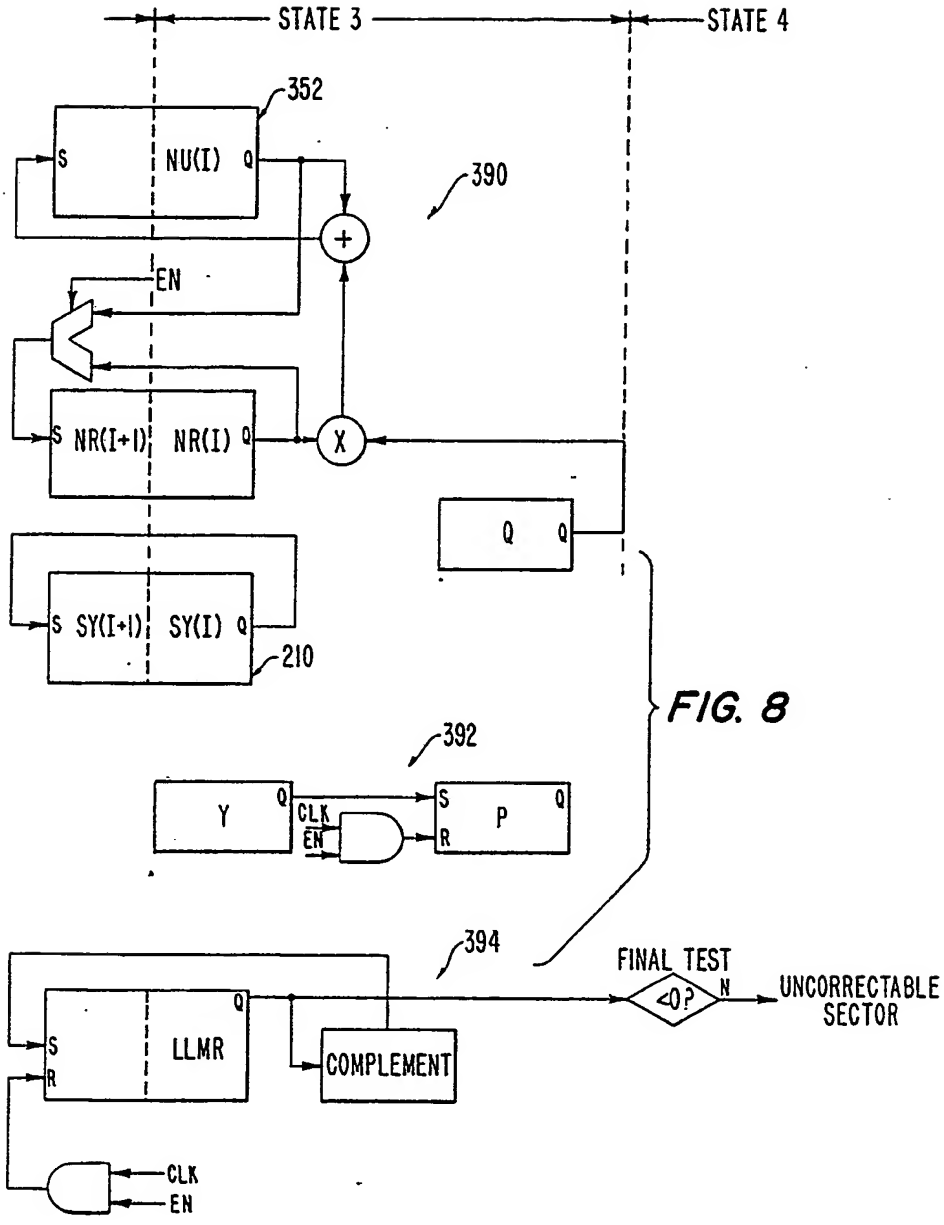

STATE 3
STATE 4
352
S
NU(I)
Q
390
+
EN
S
NR(I+1)
NR(I)
Q
X
Q
Q
S
SY(I+1)
SY(I)
Q
210
392
Y
Q
CLK
EN
S
R
P
Q
394
FINAL TEST
<0?
N
UNCORRECTABLE SECTOR
S
R
LLMR
Q
COMPLEMENT
CLK
EN

FIG. 8

FIG. 9

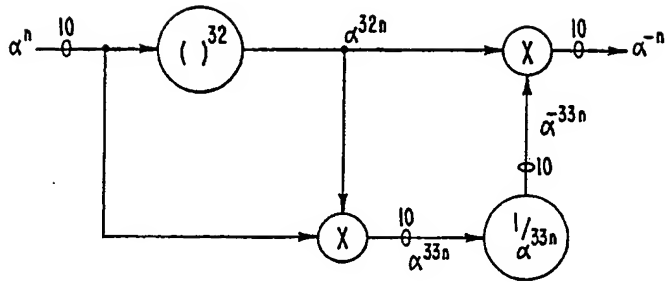

α^n
10
( )^{32}
α^{32n}
X
10
α^{-n}
α^{-33n}
10
X
10
α^{33n}
$1/\alpha^{33n}$

FIG. 10

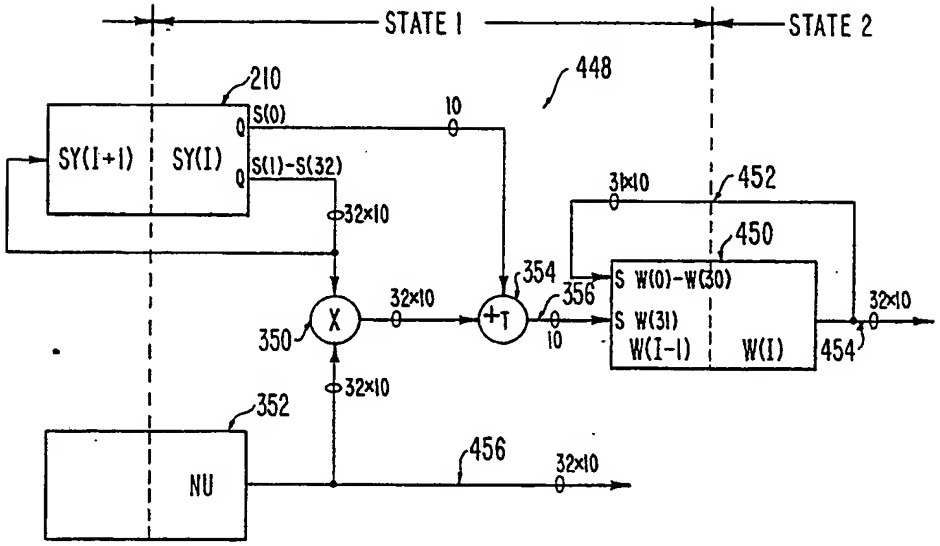

STATE 1
STATE 2
448
210
S(0)
10
SY(I+1)
SY(I)
S(1)-S(32)
32×10
31×10
452
450
354
356
S W(0)-W(30)
S W(31)
W(I-1)
W(I)
454
32×10
350
X
32×10
+T
10
32×10
352
NU
456
32×10

FIG. 11

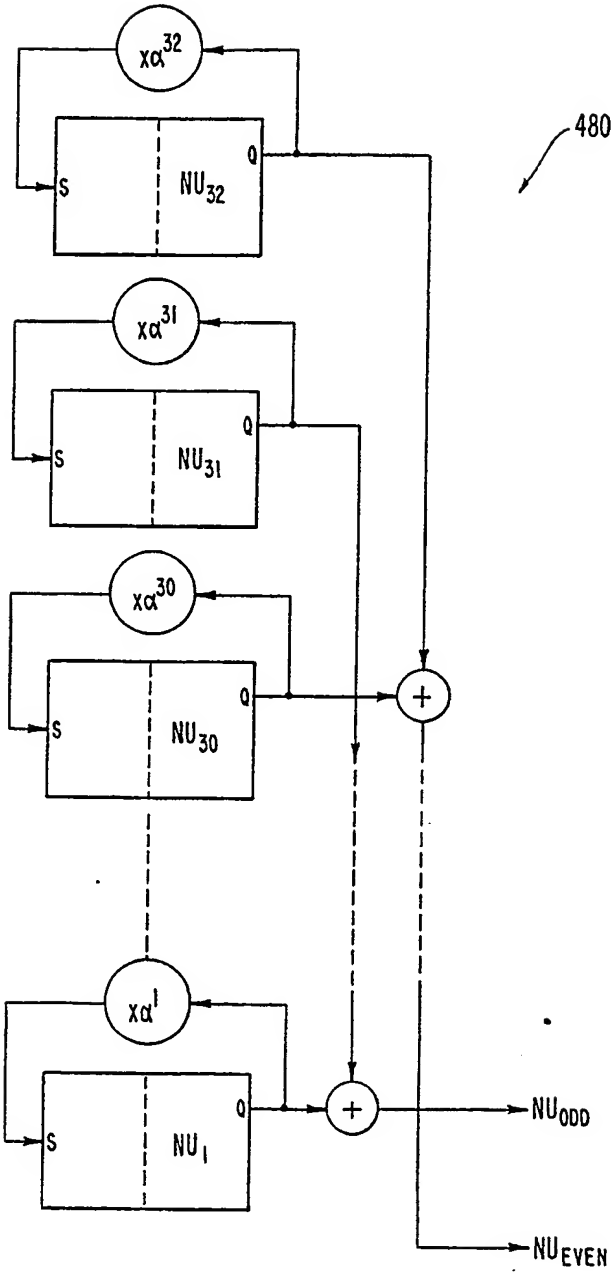
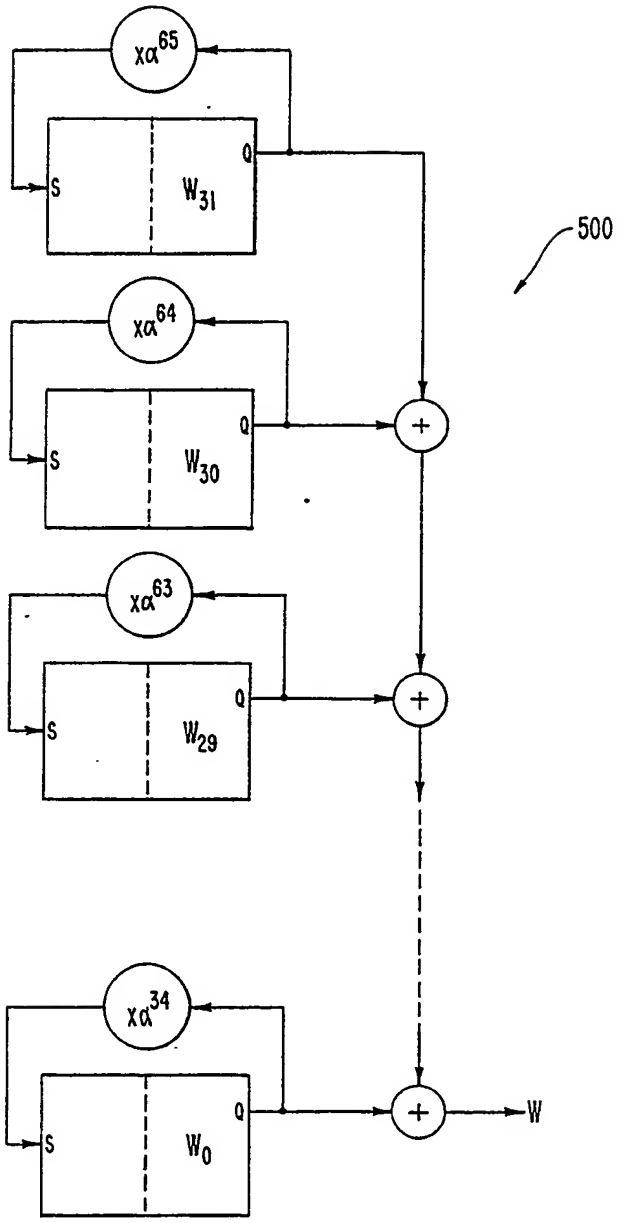

$x\alpha^{32}$
480
S
Q
NU_{32}
$x\alpha^{31}$
S
Q
NU_{31}
$x\alpha^{30}$
S
Q
NU_{30}
+
$x\alpha^{1}$
S
Q
NU_1
+
NU_{ODD}
NU_{EVEN}

FIG. 12



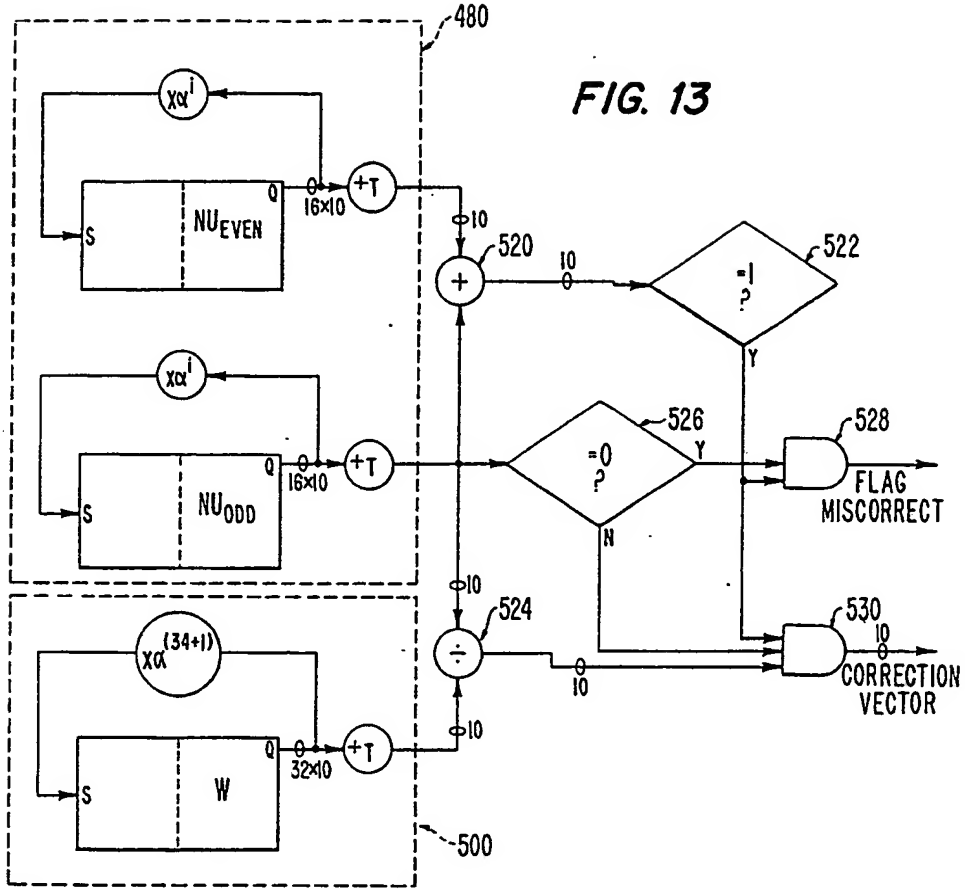

480
$X\alpha^i$
S
NU_{EVEN}
Q
16x10
+T
10
520
+
10
522
=1
?
Y
$X\alpha^i$
S
NU_{ODD}
Q
16x10
+T
526
=0
?
Y
N
528
FLAG
MISCORRECT
10
524
÷
10
530
10
CORRECTION
VECTOR
$X\alpha^{(34+1)}$
S
W
Q
32x10
+T
10
500

FIG. 13

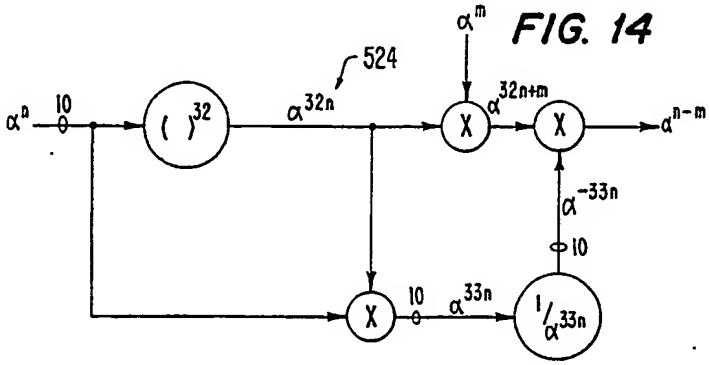
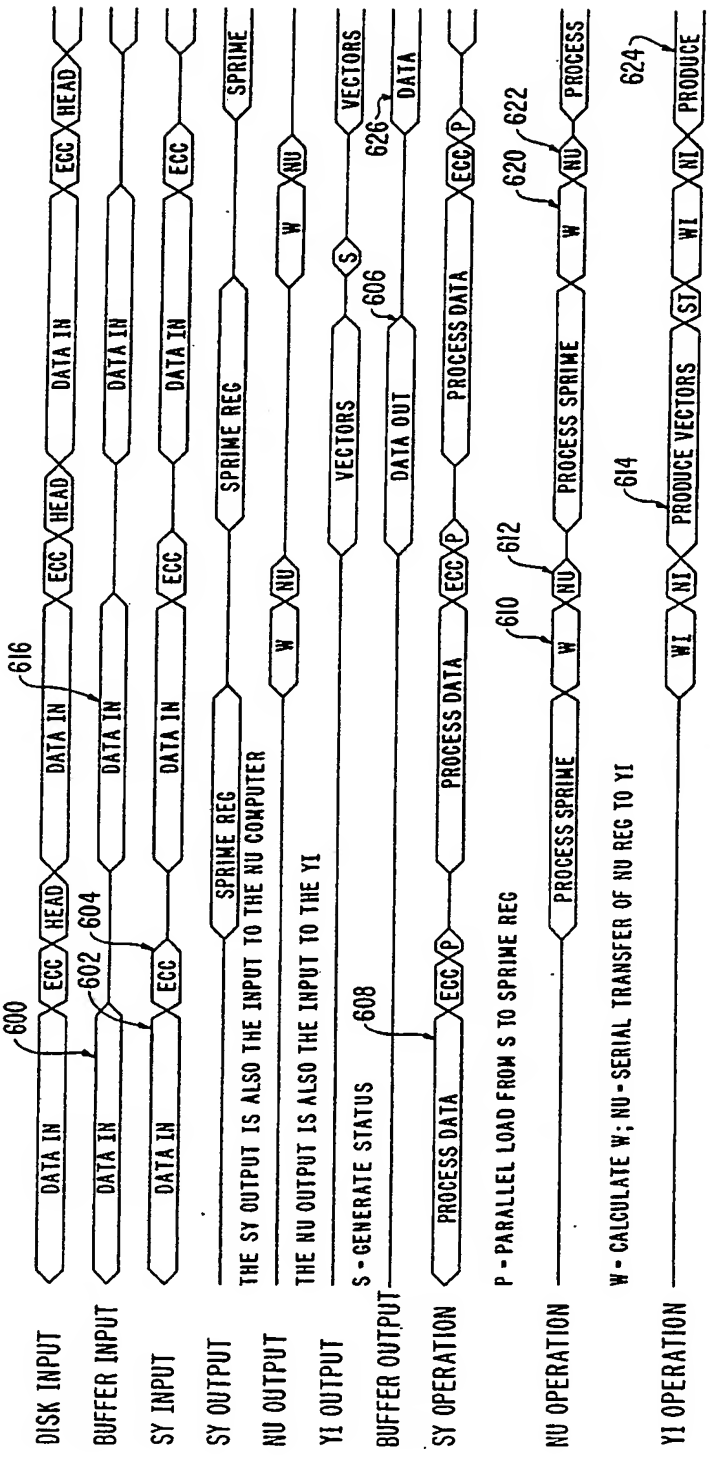

α^m
524
α^n
10
( )32
α^{32n}
X
α^{32n+m}
X
α^{n-m}
α^{-33n}
10
X
10
α^{33n}
$1/\alpha^{33n}$

FIG. 14

FIG. 15



SUBSTITUTE SHEET

FIG. 16

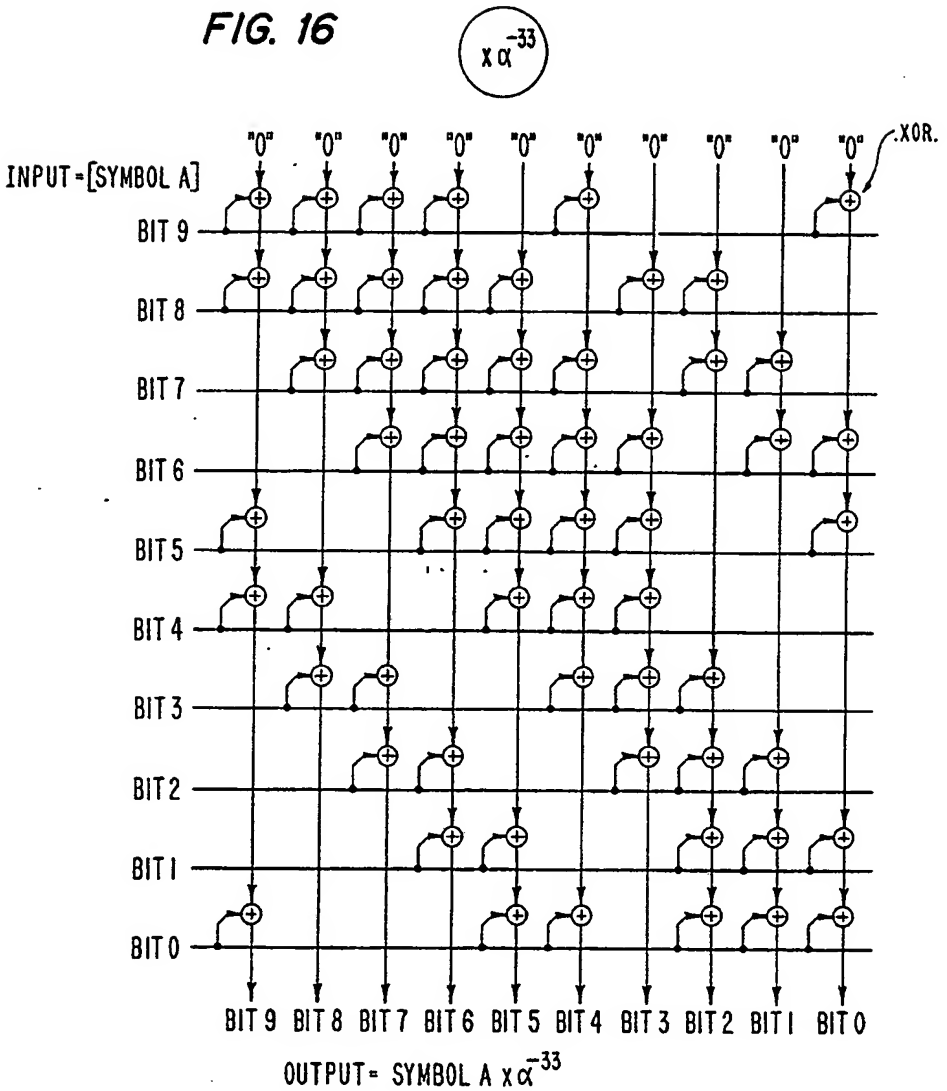

$x\alpha^{-33}$
"0" "0" "0" "0" "0" "0" "0" "0" "0" "0"
XOR.
INPUT=[SYMBOL A]
BIT 9
BIT 8
BIT 7
BIT 6
BIT 5
BIT 4
BIT 3
BIT 2
BIT 1
BIT 0
BIT 9 BIT 8 BIT 7 BIT 6 BIT 5 BIT 4 BIT 3 BIT 2 BIT 1 BIT 0
OUTPUT= SYMBOL A $x\alpha^{-33}$

FIG. 17

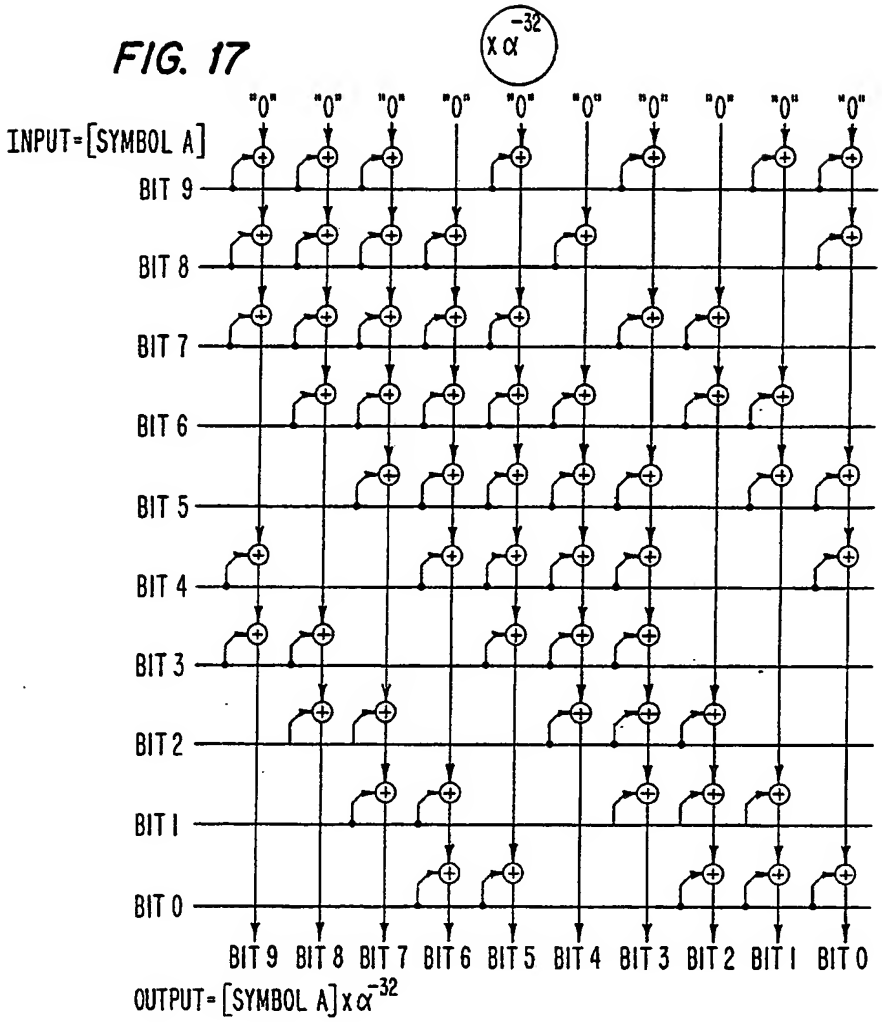

$X\alpha^{-32}$
"0" "0" "0" "0" "0" "0" "0" "0" "0" "0"
INPUT=[SYMBOL A]
BIT 9
BIT 8
BIT 7
BIT 6
BIT 5
BIT 4
BIT 3
BIT 2
BIT 1
BIT 0
BIT 9 BIT 8 BIT 7 BIT 6 BIT 5 BIT 4 BIT 3 BIT 2 BIT 1 BIT 0
OUTPUT=[SYMBOL A]$X\alpha^{-32}$

FIG. 18

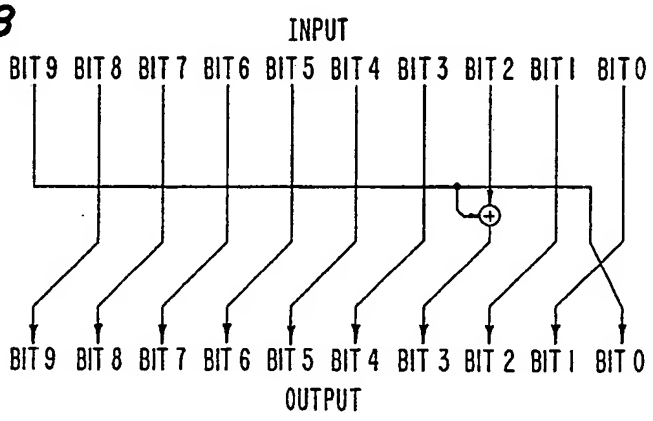

INPUT
BIT 9 BIT 8 BIT 7 BIT 6 BIT 5 BIT 4 BIT 3 BIT 2 BIT 1 BIT 0
BIT 9 BIT 8 BIT 7 BIT 6 BIT 5 BIT 4 BIT 3 BIT 2 BIT 1 BIT 0
OUTPUT

FIG. 19

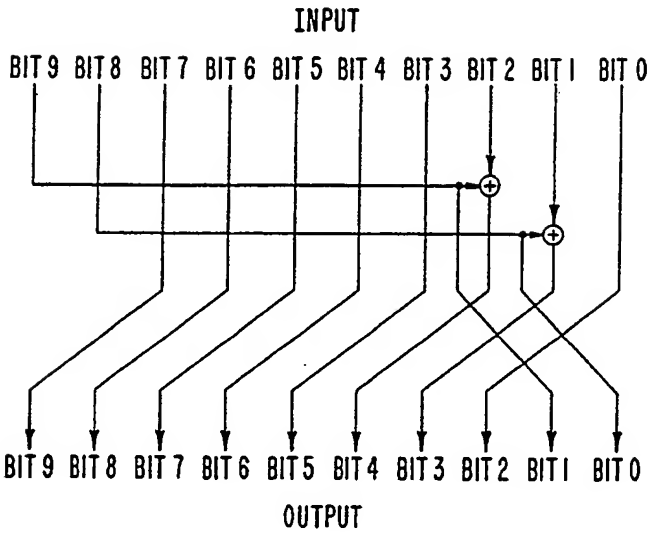


FIG. 20A

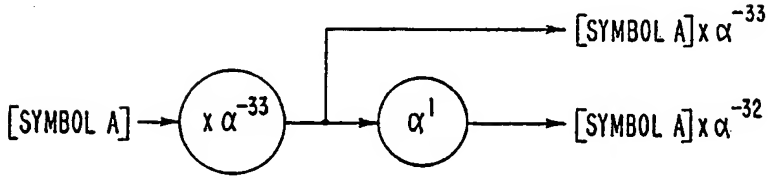


FIG. 20B

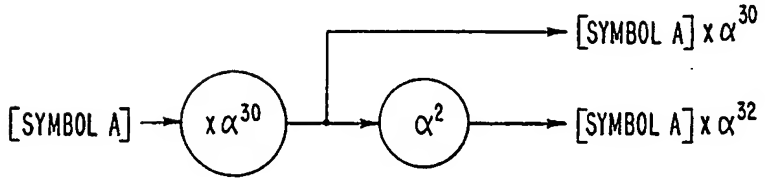


FIG. 21

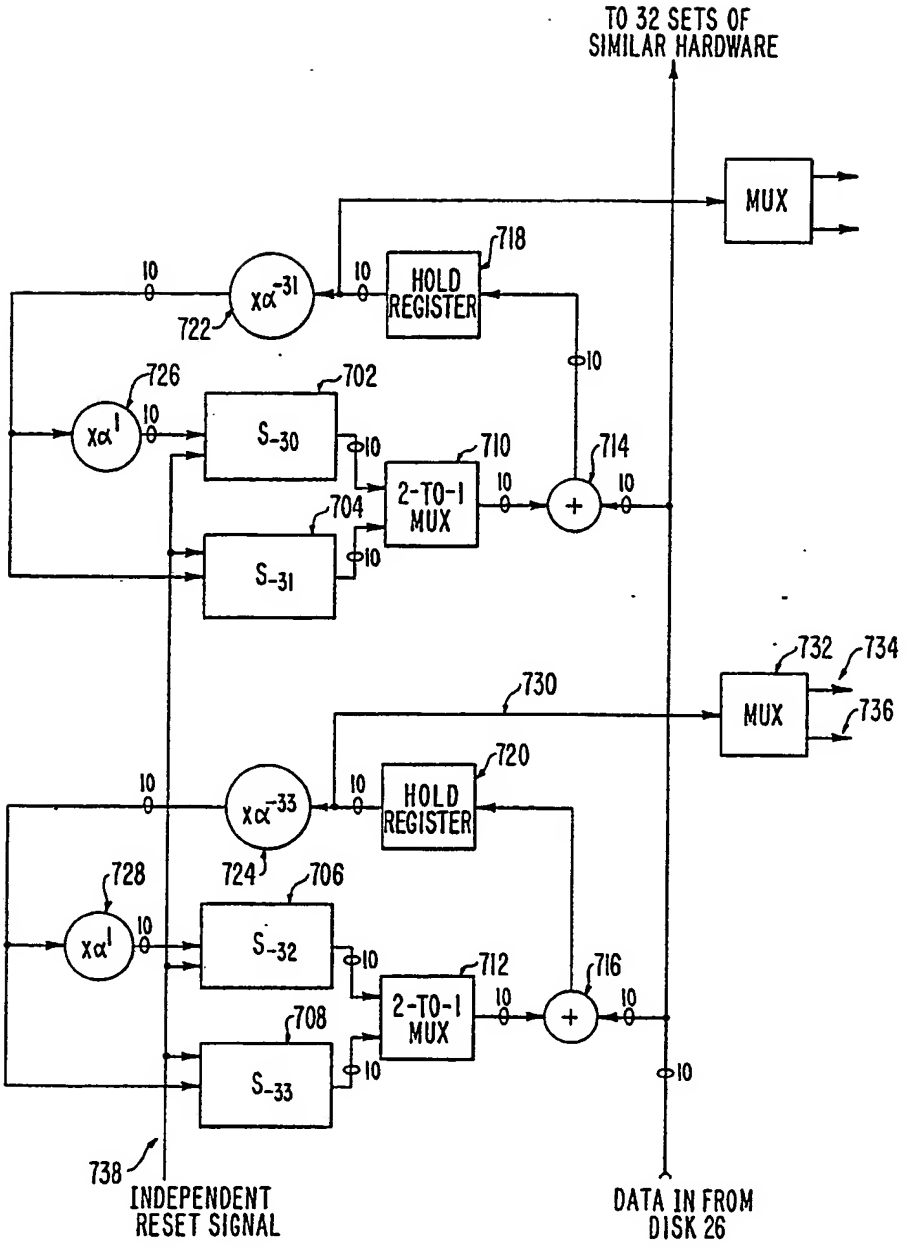

TO 32 SETS OF SIMILAR HARDWARE
MUX
718
HOLD REGISTER
10
$x\alpha^{-31}$
722
726
$x\alpha^{1}$
702
S-30
704
S-31
710
2-TO-1 MUX
714
732
734
730
736
MUX
720
HOLD REGISTER
$x\alpha^{-33}$
724
728
$x\alpha^{1}$
706
S-32
708
S-33
712
2-TO-1 MUX
716
738
INDEPENDENT RESET SIGNAL
DATA IN FROM DISK 26

FIG. 22A

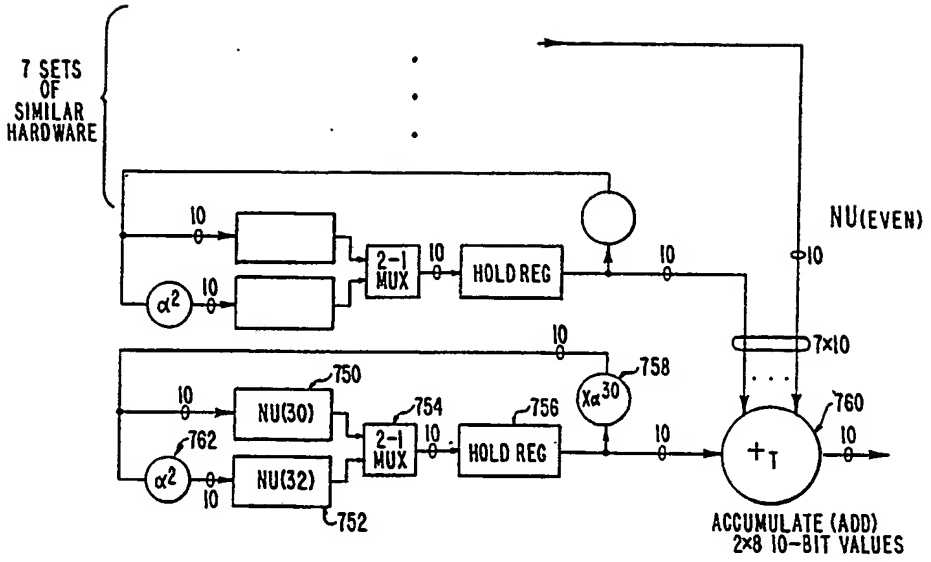


FIG. 22B

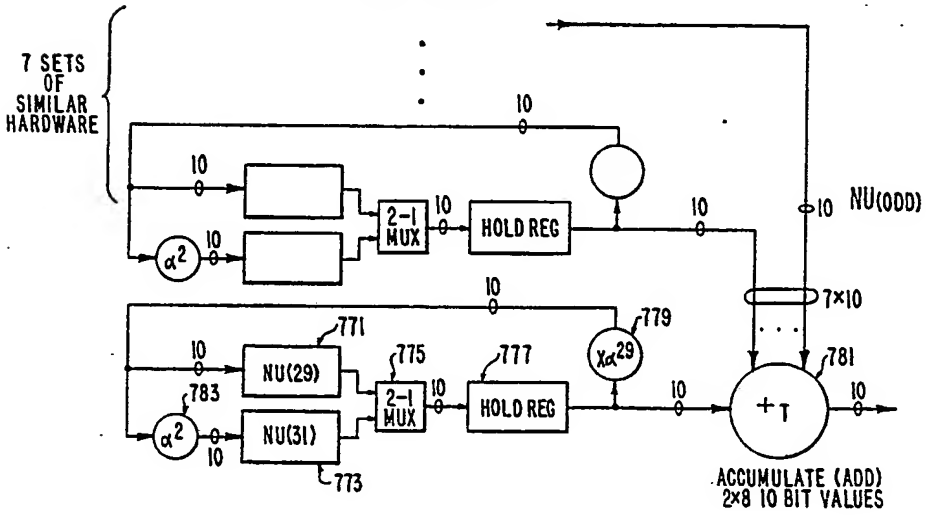
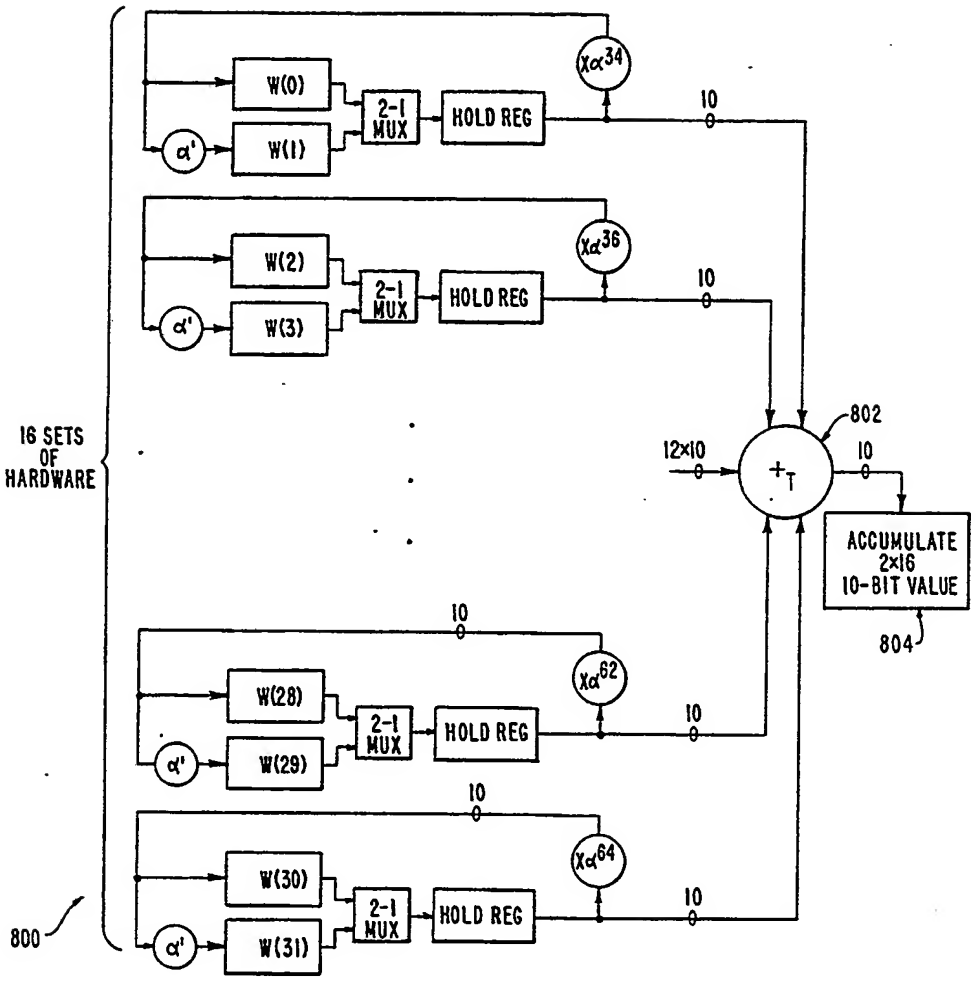


FIG. 23


16 SETS OF HARDWARE
800
W(0)
W(1)
α'
2-1 MUX
HOLD REG
Xα34
10
W(2)
W(3)
Xα36
W(28)
W(29)
Xα62
W(30)
W(31)
Xα64
12x10
+T
802
ACCUMULATE 2x16 10-BIT VALUE
804

INTERNATIONAL SEARCH REPORT

International Application No. PCT/US88/02898

I. CLASSIFICATION OF SUBJECT MATTER (if several classification symbols apply, indicate all) [6]

According to International Patent Classification (IPC) or to both National Classification and IPC

IPC(4): G06F 11/10, H03M 13/02
US Cl.: 371/37,38

II. FIELDS SEARCHED

Minimum Documentation Searched [7]

Classification System	Classification Symbols
US Cl.	371/37,38,39,40

Documentation Searched other than Minimum Documentation to the Extent that such Documents are Included in the Fields Searched [8]

III. DOCUMENTS CONSIDERED TO BE RELEVANT [9]

Category *	Citation of Document, [11] with indication, where appropriate, of the relevant passages [12]	Relevant to Claim No. [13]
X Y	IEEE Transactions on Computers, Volume C-33, Number 2, February 1984, K.Y. Liu, "Architecture for VLSI Design of Reed-Solomon Decoders", pages 178-189.	1-4,6-11, 13-28,34,35, 43,44 5
Y	R.E. BLAHUT "Theory and Practice of Error Control Codes". See pages 51-53,79,174. May 1984	5
A	US, A, 3,668,631 (GRIFFITH ET AL.) 06 June 1972. See column 4, lines 17-22.	
A	US, A, 4,099,160 (FLAGG) 04 July 1978. See Figure 1 and column 2, lines 1-9.	
A	US, A, 4,410,989 (BERLEKAMP) 18 October 1983 See column 6, line 34 to column 7, line 13.	

* Special categories of cited documents: [10]

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

IV. CERTIFICATION

Date of the Actual Completion of the International Search	Date of Mailing of this International Search Report
01 DECEMBER 1988	11 JAN 1989
International Searching Authority	**Signature of Authorized Officer**
ISA/US	Stephen M. Baker

FURTHER INFORMATION CONTINUED FROM THE SECOND SHEET

V. [X] OBSERVATIONS WHERE CERTAIN CLAIMS WERE FOUND UNSEARCHABLE [1]

This international search report has not been established in respect of certain claims under Article 17(2) (a) for the following reasons:

1. [ ] Claim numbers .. , because they relate to subject matter [12] not required to be searched by this Authority, namely:

2. [ ] Claim numbers . ..., because they relate to parts of the international application that do not comply with the prescribed requirements to such an extent that no meaningful international search can be carried out [13], specifically:

3. [X] Claim numbers 12, because they are dependent claims not drafted in accordance with the second and third sentences of PCT Rule 6.4(a).

VI. [ ] OBSERVATIONS WHERE UNITY OF INVENTION IS LACKING [2]

This International Searching Authority found multiple inventions in this international application as follows:

1. [ ] As all required additional search fees were timely paid by the applicant, this international search report covers all searchable claims of the international application.

2. [ ] As only some of the required additional search fees were timely paid by the applicant, this international search report covers only those claims of the international application for which fees were paid, specifically claims:

3. [ ] No required additional search fees were timely paid by the applicant. Consequently, this international search report is restricted to the invention first mentioned in the claims; it is covered by claim numbers:

4. [ ] As all searchable claims could be searched without effort justifying an additional fee, the International Searching Authority did not invite payment of any additional fee.

Remark on Protest

[ ] The additional search fees were accompanied by applicant's protest.

[ ] No protest accompanied the payment of additional search fees.